Giáo trình Lập trình Pascal căn bản

– 1 —

BÀI 1: GIỚI THIỆU NGÔN NGỮ PASCAL VÀ CÁC VÍ DỤ ĐƠN GIẢN

I. Xuất xứ ngôn ngữ Pascal:

Pascal là ngôn ngữ lập trình cấp cao do *Niklaus Wirth*, giáo sư điện toán trường đại học kỹ thuật Zurich *(Thụy Sĩ)*, đề xuất năm 1970 với tên Pascal để kỷ niệm nhà toán học và triết học nổi tiếng *Blaise Pascal* (người Pháp).

Ngôn ngữ lập trình Pascal có 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 và được dùng để tạo ra nhiều ứng dụng trên 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 bổ sung, giảm thiểu khác nhau. Ví dụ: *TURBO PASCAL* của hãng Borland *(Mỹ), QUICK PASCAL* của hãng Microsoft, *UCSD PASCAL (**U**niversity of **C**alifornia at **S**an **D**iego),* ANSI PASCAL *(**A**merican **N**ational **S**tandard **I**nstitute),.v.v.*

So với nhiều sản phẩm Pascal của nhiều tổ chức và công ty khác nhau xuất bản, *TURBO PASCAL* của hãng Borland tỏ ra có nhiều ưu điểm nhất và hiện nay đã trở thành ngôn ngữ lập trình phổ biến nhất trên thế giới sử dụng trong lĩnh vực giảng dạy và lập trình chuyên nghiệp. Chỉ trong vòng vài 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*** *(1972).*

Các tập tin chính của ngôn ngữ Turbo Pascal gồm:

- *Turbo.exe:* chương trình soạn thảo, dịch và liên kết chương trình.
- *Turbo.tpl (.tpl - Turbo Pascal Library):* tập tin thư viện lưu các đơn vị *(Unit)* chuẩn để chạy với Turbo.exe.

Muốn sử dụng các lệnh đồ họa, phải có các tập tin sau:

- *Graph.tpu:* Đơn vị *(Unit)* chứa các lệnh đồ hoạ.
- *Các tập tin có phần mở rộng CHR (SANS.CHR, TRIP.CHR, GOTH.CHR,.v.v.):* Chứa các kiểu chữ trong chế độ đồ hoạ.
- Các tập tin có phần mở rộng *BGI (EGAVGA.BGI, HERC.BGI, CGA.BGI,...):* để điều khiển các loại màn hình tương ứng khi dùng đồ hoạ.

II. Khởi động:

Ta có thể khởi động Pascal từ *Windows hoặc MS-DOS*, chuyển đến thư mục *BP* hoặc *TP* và chạy tập tin *BP.EXE* hay *TURBO.EXE*. Hai cách khởi động trên thực hiện như sau:

- *Khởi động từ dấu nhắc của MS-DOS:* Chuyển đến thư mục *BP* hoặc *TP* nơi chứa tập tin *BP.EXE* hoặc *TURBO.EXE,* gõ *BP* hoặc *TURBO* và ấn <Enter>.
- *Khởi động từ Windows:* chọn menu *Start/Program/Borland Pascal*. Nếu chương trình Pascal chưa được cài vào menu *Start*, bạn có thể dùng *Windows Explorer* chuyển đến tập tin *BP.EXE* hoặc *TURBO.EXE* và khởi động Pascal bằng cách chạy tập tin này.

III. Các phím chức năng cần biết của ngôn ngữ Pascal:

- ***F2:*** Lưu chương trình trong khi soạn thảo.
- ***F3:*** Tạo một file mới hoặc mở một file cũ.
- ***F9:*** Dịch thử chương trình để kiểm tra lỗi.
- ***Ctrl - F9:*** Chạy chương trình.
- ***Alt - F5:*** Xem kết quả chạy chương trình.
- ***Alt - X:*** Thoát khỏi màn hình soạn thảo chương trình Pascal.

IV. Cấu trúc một chương trình Pascal:

1. Cấu trúc cơ bản:

Chương trình Pascal đơn giản nhất phải có hai từ khoá *Begin* và *End* như sau:

```
Begin


End.
```

Chương trình trên tuy không làm gì khi chạy *(ấn Ctrl - F9)* nhưng là một chương trình hợp lệ do hội đủ điều kiện cần thiết là có hai từ khoá *Begin* và *End.*

Từ khoá *End* có kèm dấu "." phía sau báo hiệu kết thúc chương trình, đây là điều bắt buộc phải có trong một chương trình. Từ khoá *Begin* trên được trình biên dịch hiểu là bắt đầu thực hiện các lệnh sau nó và kết thúc tại từ khoá *End* có dấu chấm ".". Khối lệnh nằm trong cặp từ khoá *Begin* và *End* nếu có dấu chấm theo sau còn gọi là khối chương trình chính. Ngoài ra, nếu sau từ khoá *End* không có dấu hoặc có dấu ";" thì đó có thể là khối chương trình con, khối lệnh của hàm hoặc khối lệnh trong chương trình. Trong chương trình có thể có nhiều khối lệnh, tức có thể có nhiều cặp từ khoá *Begin* và *End.*

2. Phương pháp khai báo và tổ chức cấu trúc một chương trình Pascal:

Việc đặt các phần khai báo và soạn thảo chương trình theo thứ tự như sau:

```
Program ProgName;
Uses UnitName1, UnitName2, UnitNameN;
Label LabelName1, LabelName2, LabelNameN;
Const Const1 = n, Const2 = m, ConstN = k;
Type Type1 = AnyType;
Var Var1, Var2, VarN : Type;
Begin

    { Các lệnh của chương trình }

End.
```

Ö Giải thích cấu trúc các khai báo trên:

Nếu có phần khai báo nào cần cho chương trình thì phải tuân theo thứ tự trên, ví dụ: phần khai báo thư viện *(USES)* không thể đặt sau phần khai báo hằng số *(CONST)* hoặc sau *(VAR)...* sau mỗi phần khai báo phải có dấu ';'.

- *Program:* Từ khoá này dùng để khai báo tên chương trình, *ProgName* là tên chương trình, tên này khác với tên tập tin. Tên chương trình phải tuân theo quy tắc:
  + không có ký tự trống xen giữa.
  + không đặt số ở ký tự đầu tiên.
  + trong phần tên không chứa các ký tự đặt biệt như: *'!', '@', '#', '$', '%', '^', '&', '* ', '(', ')', '-', '+', '/', '|', '.', ',',.v.v.*
  + kết thúc phải có dấu '**;**'.
  + phần này có thể không có.

4 Ví dụ: một cách khai báo tên chương trình:

```
Program TimUSCLN;
Begin
        ...
End.
```

- *Uses:* Từ khoá này dùng để khai báo việc sử dụng *Unit (thư viện)* cho chương trình. Thư viện là tập hợp các hàm, thủ tục *do ngôn ngữ Pascal cung cấp kèm theo hoặc cũng có thể do người lập trình tạo ra để sử dụng.* Ta khai báo thư viện thông qua tên của thư viện, và trong chương trình đó ta sẽ có thể sử dụng các thủ tục hoặc

các hàm có trong thư viện đó. Các thư viện chuẩn của ngôn ngữ Pascal gồm: *CRT, DOS, GRAPH, GRAPH3, OVERLAY, PRINTER, SYSTEM* và *TURBO3*. Trong đó, thư viện *SYSTEM* mặc định được chuyển vào chương trình mà ta không cần phải khai báo. Ví dụ một cách khai báo thư viện:

```
...
Uses  CRT, GRAPH;
...
```

- *Label:* Dùng để khai báo các nhãn cho chương trình. Nhãn là các tên dùng để đánh dấu trong chương trình để lệnh *GOTO* nhảy đến đúng vị trí đó. Việc sử dụng lệnh *GOTO* được đề cập ở bài 4. Ví dụ một cách khai báo nhãn:

```
...
Label  TH1, N2;
...
```

- *Const:* Từ khoá này dùng để khai báo các hằng số sử dụng trong chương trình, khi báo hằng số là việc cố định một vài giá trị nào đó trong chương trình thông qua tên hằng, ví dụ cách khai báo hằng:

```
...
Const  k = 5, Max = 500, Ten = 'Nam';
...
```

- *Type:* từ khoá dùng để khai báo các kiểu hằng dữ liệu sử dụng cho chương trình. Với từ khoá này, ta có thể tự tạo riêng cho mình những kiểu dữ liệu riêng dựa trên các kiểu dữ liệu chuẩn để tiện sử dụng trong việc lập trình. Các khái niệm về dữ liệu chuẩn và phương pháp tạo kiểu dữ liệu tự tạo sẽ được giới thiệu ở các phần sau. Ví dụ một cách để khai báo một kiểu dữ liệu tự tạo:

```
...
Type  Day = Array [1..7] of  String[8];
...
```

- *Var:* Từ khoá dùng để khai báo các biến số được sử dụng trong chương trình. Biến số là các *giá trị có thể thay đổi được* trong suốt quá trình chạy của chương trình. Khái niệm về biến số rất quan trọng trong việc lập trình *(khái niệm này được trình bày kỹ ở bài 3).* Một ví dụ về cách khai báo biến:

```
...
```

```
Var   HoDem, Ten : String;
                N  : Integer;
...
```

Ö Ghi chú:

- Thứ tự các khai báo trên là điều bắt buộc, ta phải nắm thứ tự này cho dù một số khái niệm ta chưa được biết.

- Trong chương trình Pascal, để tạo lời chú thích, ta sử dụng cặp dấu {...} hoặc (*...*) lồng các câu chú thích vào bên trong nó.

- Trên một dòng có thể viết một hoặc nhiều câu lệnh.

V. Các ví dụ đơn giản làm quen với ngôn ngữ Pascal:

4 Ví dụ 1:

```
Program  GioiThieu;
Begin
   Writeln ( ' Trung tam Trung hoc Chuyen nghiep va Day nghe ' );
   Write   ( '        74 Tran Quoc Toan - Tel: 0511 872664        ' );
End.
```

F *Giải thích chương trình GioiThieu:*

- *Begin:* Từ khoá cho biết bắt đầu chương trình.

- *Writeln:* là thủ tục xuất nội dung các thành phần bên trong cặp dấu *(...)* lên màn hình và chuyển con trỏ xuống dòng. Bên trong cặp dấu *(...)* có thể có nhiều thành phần gồm *chuỗi ký tự (hằng giá trị chuỗi), biến số* hoặc *hàm*. Giữa các thành phần trong cặp dấu *(...)* phải cách nhau bằng dấu ',' nếu không cùng loại, tức là chuỗi ký tự phải được cách với biến số hoặc hàm đứng trước nó hay sau nó bằng dấu ','. Chuỗi ký tự muốn hiển thị nguyên văn phải được đặt trong cặp dấu ' '.

- *Write:* là thủ tục xuất nội dung các thành phần bên trong cặp dấu *(...)* lên màn hình, thủ tục này có chức năng tương tự *Writeln* nhưng không chuyển con trỏ xuống dòng.

- *End:* là từ khoá cho biết kết thúc chương trình.

- Các dòng lệnh nằm giữa *Begin* và *End* là lệnh mà chương trình cần phải thực hiện.

- Để xem chương trình trên, ta chạy bằng *Ctrl - F9* và xem lại bằng *Alt - F5*.

4 Ví dụ 2:

```
Program  DonXinPhep;
Uses  CRT;
Begin
   ClrScr;
   Writeln ( ' ********************************** ' );
   Writeln ( ' * Cong hoa Xa hoi Chu nghia Viet Nam * ' );
   Writeln ( ' *       Doc Lap - Tu Do - Hanh Phuc       * ' );
   Writeln ( ' *       DON XIN PHEP NGHI HOC        * ' );
   Writeln ( ' ********************************** ' );
   Writeln ( '... ' );
   Readln;
End.
```

F Giải thích chương trình trên:

- Khai báo: *Uses CRT;* **ð** khai báo thư viện *CRT*, do có sử dụng lệnh *ClrScr.*

- Lệnh *ClrScr;* **ð** lau sạch màn hình *(Clear Screen).*

- Các lệnh *Writeln (...)* **ð** xuất ra màn hình nội dung bên trong dấu *(...)* và xuống dòng.

- Lệnh *Readln;* **ð** dừng chương trình, phương pháp này dùng để *hiển thị nội dung sau khi thực hiện các lệnh bên trên và chờ người dùng ấn phím bất kỳ để tiếp tục thực hiện các lệnh kế sau nó.* Trong trường hợp trên, kế tiếp là từ khoá *End* nên chương trình được kết thúc sau khi có một phím bất kỳ được ấn.

4 Ví dụ 3:

```
Program  TinhTong;
Uses  CRT;
Begin
   ClrScr;
   Write ( ' 30 + 40 + 15 = ', 30 + 40 + 15 );
   Readln;
End.
```

1 Kết quả: Máy thực hiện phép tính và hiển thị *30 + 40 + 15 = 85*

F Trong câu lệnh Write ở trên, có hai thành phần, biểu thức thứ nhất: *'30 + 40 +15 = '* được hiểu là một chuỗi phải được hiển thị nguyên văn do có cặp dấu ' ' ở hai đầu. Thành phần thứ hai được cách với thành phần thứ nhất bằng dấu ',' và do không có cặp dấu ' ' hai đầu nên nó được tính tổng và trả về giá trị của biểu thức.

_____ o ² o _____

BÀI 2 : CÁC KHÁI NIỆM CƠ BẢN CỦA NGÔN NGỮ PASCAL

I. Các từ khoá *(Key word)* trong ngôn ngữ Pascal:

Các từ khoá là các từ dùng để khai báo, đặt tên cho đối tượng trong Pascal, khi ta đặt tên cho đối tượng nào đó, không được đặt trùng tên với các từ khoá.

Bảng từ khoá trong ngôn ngữ Pascal gồm:

and, array, asm, begin, case, const, constructor, destructor, div, do, downto, else, end, file, for, function, goto, if, implementation, in, inline, interface, label, mod, nil, not, object, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor.

Turbo Pascal không phân biệt ký tự thường hoặc hoa. Ví dụ, các cách viết sau có ý nghĩa như nhau: *Begin, BEGIN, begin, beGIN, bEGIN,...*

II. Các kiểu dữ liệu cơ bản:

1. Các kiểu dữ liệu dạng số nguyên:

a. Kiểu *Byte:* Kiểu *Byte* thuộc kiểu dữ liệu biểu diễn các giá trị số nguyên từ ***0*** đến ***255***. Kiểu Byte chiếm 1 byte trên bộ nhớ.

b. Kiểu *Integer:* Kiểu *Integer* là kiểu dữ liệu biểu diễn các giá trị số nguyên từ ***-32768*** đến ***32767***. Kiểu Integer chiếm 2 bytes trên bộ nhớ.

c. Kiểu *Shortint:* Kiểu *Shortint* là kiểu dữ liệu biểu diễn các giá trị số nguyên từ ***-128*** đến ***127***. Kiểu Shortint chiếm 1 byte trên bộ nhớ.

d. Kiểu *Word:* Kiểu Word là kiểu dữ liệu biểu diễn các giá trị nguyên từ ***0*** đến ***65535***. Kiểu *Word* là kiểu số *không biểu diễn được giá trị âm*. Kiểu Word chiếm 2 bytes trên bộ nhớ.

e. Kiểu *Longint:* Kiểu *Longint* biểu diễn các giá trị số nguyên từ ***-2.147.483.648*** đến ***2.147.483.647***. Kiểu Longint chiếm 4 bytes trên bộ nhớ.

2. Các kiểu dữ liệu dạng số có phần biểu diễn thập phân:

a. Kiểu *Single:* Là tập hợp các số theo kiểu dấu '.' động trong giới hạn từ ***1.5E -45*** đến ***3.4 E38*** *(1,5 x 10^{-45} đến 3,4 x 10^{38}).* Kiểu Single chiếm 4 bytes trên bộ nhớ.

b. Kiểu *Real:* Là tập hợp các số theo kiểu dấu '.' động trong giới hạn từ ***2.9E -39*** đến ***1.7E 38*** *(2,9 x10^{-39} đến 1,7 x 10^{38}).* Kiểu Real chiếm 6 bytes trên bộ nhớ.

c. Kiểu *Double:* Là tập hợp các số theo kiểu dấu ',' động trong giới hạn từ ***5.0E -324*** đến ***1.7E 308*** *(5,0 x10 $^{-324}$ đến 1,7 x 10 ^{308}).* Kiểu Double chiếm 8 bytes trên bộ nhớ.

3. Kiểu Char *(ký tự)*:

Kiểu *Char* dùng để biểu diễn các giá trị là các ký tự thuộc bảng chữ cái: *'A', 'b', 'x',...* các con số: *0..9* hoặc các ký tự đặc biệt : *'!', '@', '#', '$', '%', '&', '*',...*

Để biểu diễn thông tin, ta cần phải sắp xếp các ký tự theo một chuẩn nào đó và mỗi cách sắp xếp đó gọi là bảng mã, thông dụng nhất là bảng mã *ASCII (American Standard Code for Information Interchange).* Bảng mã *ASCII* có *256* ký tự được gán mã số tư *0..255,* mỗi ký tự có một mã số nhất định, ví dụ : ký tự *'A'* có mã số là *65*, *'a'* có mã số là *97* trong bảng mã *ASCII,.v.v.*

Để hiển thị bảng mã ASCII, bạn chạy chương trình sau:

```
Program  ASCI I_Table;
Uses  CRT;
Var  I : Integer;
Begin
   ClrScr;
   For  I := 0 to 255 do
     Write( I, ' = ' , CHR( I ), '    ' );
   Readln;
End.
```

4. Kiểu Logic:

Kiểu logic là kiểu biểu diễn hai trạng thái là đúng *(True)* hoặc sai *(False).* Từ khoá để khai báo cho kiểu logic là *BOOLEAN.*

4 Ví dụ:

```
Var  Co : Boolean;
     Co := True;
```

5. Kiểu String *(chuỗi ký tự)*:

String là kiểu dữ liệu chứa các giá trị là nhóm các ký tự hoặc chỉ một ký tự, kể cả chuỗi rỗng. Độ dài tối đa của một biến kiểu String là 255, tức là nó có thể chứa tối đa một dãy gồm 255 ký tự.

Cú pháp khai báo: *(1) Var Biến_1, Biến_2, Biến_n: String;*

Hoặc *(2) Var Biến_1, Biến_2, Biến_n: String [30];*

Cách khai báo *(1)* sẽ cho phép biến *HoTen* nhận tối đa 255 ký tự. Cách *(2)* cho phép biến *HoTen* nhận tối đa *30 ký tự.*

Ö Ghi chú: *Cách sử dụng kiểu dữ liệu String sẽ được trình bày chi tiết ở bài 8.*

III. Các hàm xử lý dữ liệu cơ bản của ngôn ngữ Pascal:

- ***SQR(x)*** bình phương của một số nguyên hay thực.
- ***ABS(x)*** trị tuyệt đối của x.
- ***SQRT(x)*** căn bậc hai của x.
- ***SIN(x)*** tính giá trị Sin(x) với x là Radian.
- ***COS(x)*** tính giá trị Cos(x) với x là Radian.
- ***ARCTAN(x)*** tính giá trị Arctan(x).
- ***LN(x)*** hàm logaric cơ số e = 2.718.
- ***EXP(x)*** hàm e^x.
- ***TRUNC(x)*** cắt bỏ phần thập phân của x nếu có. Ví dụ: *Trunc(4.86) = 4, Trunc(-3.2) = 4.*
- ***ROUND(x)*** cho số nguyên gần x nhất. Ví dụ: *Round(1.6) = 2, Round(-23.68) = -24, Round(1.5) = 2.*
- ***PRED(x)*** cho giá trị đứng trước x, đối số x có thể là kiểu logic, kiểu nguyên hoặc kiểu ký tự. Ví dụ: *Pred('B');* ð cho giá trị *'A'*, *Pred(2)* cho giá trị *1*, *Pred(True)* cho giá trị *False*. Tuy nhiên, *Pred(False)* lại không cho được giá trị nào do *giá trị False đứng trước giá trị True đối với kiểu Boolean.*
- ***SUCC(x)*** cho giá trị đứng sau x, đối số x có thể là kiểu logic, kiểu nguyên hoặc kiểu ký tự. Ví dụ: *Succ('B');* ð cho giá trị *'C'*, *Succ(2)* cho giá trị *3*, *Succ(False)* cho giá trị *True*.
- ***ORD(x)*** cho số thứ tự của ký tự x trong bảng mã *ASCII*. Ví dụ: *Ord('A') = 65, Ord('a') = 97,...*
- ***CHR(x)*** trả về ký tự thứ x trong bảng mã *ASCII*. Ví dụ: *Chr(65) = 'A', Chr(50) = 2,...*

- *ODD(x)* Trả về giá trị *True* nếu x là số lẻ và trả về giá trị *False* nếu x là số chẵn.

IV. Sử dụng hàm Random(n) để lấy một giá trị nguyên ngẫu nhiên:

Hàm *Random(n)* sẽ trả về một giá trị nguyên mà máy lấy ngẫu nhiên có giá trị từ 0 đến n. Trong đó, *n là một số kiểu Word* tức là trong khoản từ *0.. 65535.*

Trước khi sử dụng hàm Random ta phải gọi thủ tục *Randomize* để khởi tạo bộ tạo số ngẫu nhiên

_____ o ² o _____

BÀI 3: HẰNG SỐ, BIẾN SỐ, BIỂU THỨC VÀ CÂU LỆNH ĐƠN GIẢN TRONG NGÔN NGỮ PASCAL

I. Hằng số:

1. Khái niệm:

- Hằng số là các giá trị không thay đổi trong quá trình chạy chương trình.
- Có hai phương pháp sử dụng hằng :

+ Gán trực tiếp giá trị hằng.Ví dụ: *DT := R * R * 3.14; ChuVi := D * 3.14;*

+ Đặt cho hằng một tên gọi và trong quá trình soạn chương trình ta dùng tên gọi thay cho việc dùng trực tiếp giá trị đó. Ví dụ: *ChuVi := D * Pi;* trong đó, *Pi* là một hằng số chuẩn của Pascal *(tức là ta có thể dùng mà không cần khai báo và gán giá trị).*

- Hằng số luôn luôn được khai báo trước phần khai báo biến nếu sử dụng theo phương pháp đặt tên cho hằng.

2. Cú pháp khai báo:

Const a_1 *= Trị_số_1,* a_2 *= Trị_số_2,* a_n *= Trị_số_n;*

Trong đó: a_1... a_n là tên các hằng số, các *trị_số_1,2,...,n* là các giá trị gán cho các tên hằng a_1...a_n.

F Ví dụ một cách khai báo hằng số: *Const Pi = 3.1416, Max = 500;*

4 Ví dụ: chương trình tính chu vi đường tròn có sử dụng hằng số *Pi* do ta định nghĩa:

```
Program TinhCV_DT_HT;
Const Pi = 3.1416;
Var R :Real;
Begin
```

```
    Write ( ' Nhap ban kinh hinh tron : ' );
    Readln (R);
    Writeln ( ' Dien tich hinh tron =  ' , Pi * R * R );
    Writeln ( ' Chu vi hinh tron =  ' , 2 * R * Pi);
    Readln;
End.
```

Ö Ghi chú:

- Ta tránh viết: *z := Exp(1.23) + Sin(2.34) * Sin(2.34);*
- Ta sẽ thấy tai hại ngay vì khi muốn tính lại *z* với giá trị mới của *x*, ví dụ *x = 1.55*, không lẽ lại đi thay hết 3 vị trí với *2.34 (là giá trị cụ thể của x mà ta đã không sử dụng hằng số)* thành *1.55* !!

- Trong chương trình trên, bạn có thể *tối ưu hoá* thêm để chương trình chạy nhanh hơn bằng cách thay hai lần tính *Sin(x)* bằng một lần. Cụ thể, ta thực hiện như sau:

```
t := Sin(x);
z := Exp(a + t * t - x);
```

Tác phong *tối ưu hoá* này sẽ rất có ích cho bạn khi bạn có một chương trình với khối lượng tính toán đồ sộ, có thể chạy vài ngày đêm liên tục nhưng nếu biết tối ưu ngay từ đầu thì sẽ giảm bớt xuống còn một ngày chẳng hạn. Lúc này bạn mới 'thấu hiểu' *tối ưu hoá* để làm gì ?

II. Biến số:

1. Khái niệm:

- Là đại lượng mà *giá trị của nó có thể thay đổi trong quá trình thực hiện chương trình*. Biến được khai báo bằng từ khoá *VAR.*
- Biến là tên của một vùng bộ nhớ lưu trữ dữ liệu.
- Biến được truy xuất trong chương trình thông qua tên biến.
- Biến là một cấu trúc ghi nhớ dữ liệu vì vậy phải được quy định theo một kiểu dữ liệu nào đó, ví dụ kiểu *Integer, Byte, Char,...*

2. Cú pháp khai báo cho các biến:

VAR Tên_biến_1, Tên_biến_2, Tên_biến_n : Kiểu_dữ_liệu_của_biến;

Trong đó: *Tên_biến_1*, Tên_biến_2, Tên_biến_*n* là *tên* các biến cần khai báo để sử dụng trong chương trình, *Kiểu_dữ_liệu_của_biến* là một trong các kiểu dữ liệu

chuẩn *(đã được nêu trong phần II của bài 2)* của Pascal hoặc do người dùng định nghĩa.

F Ví dụ một cách khai báo biến:

```
Var   a,b : Integer;
      c   : Real;
      Ten : String [10];
```

4 Ví dụ: chương trình tính tổng hai số nguyên được nhập từ bàn phím. Trong bài này, ta cần khai báo hai biến *a* và *b* để tính toán.

```
Uses  CRT;
Var  a, b : Integer;
Begin
   ClrScr;
   Write( ' Nhap so thu nhat : ' );
   Readln(a);
   Write( ' Nhap so thu hai : ' );
   Readln(b);
   Write( ' Ket qua : ', a ,' + ', b ,' = ', a + b);
   Readln;
End.
```

III. Biểu thức:

Một biểu thức được tạo bởi các *toán tử (phép toán)* và các *toán hạng* dùng để thể hiện một công thức toán học. *Toán hạng* có thể là *hằng, hàm* hoặc *biến*.

4 Ví dụ: Sau khi khai có báo:

```
Const  Max = 120;
Var    x: Integer;
```

ta có thể viết biểu thức sau: *5 + Max * Exp(x);*

Trong đó: *+* và *** là hai toán tử, các hằng số *5*, *Max* và hàm *Exp(x)* là các toán hạng.

Ö Chú ý:

- *Một hằng, một biến, một hàm cũng được xem là biểu thức, đó là biểu thức đơn giản.*
- Các phép toán trong một biểu thức được sắp xếp theo thứ tự ưu tiên như sau:

+ Các phép toán một ngôi được ưu tiên thứ nhất là: dấu dương *(+),* dấu âm *(-),* phép phủ định *(not).*

+ Các phép toán nhân chia: *nhân (*), chia (/), lấy phần nguyên (div), lấy phần dư (mod), phép và (and).*

+ Các phép cộng trừ: cộng *(+),* trừ *(-),* phép hoặc *(or).*

+ Các phép so sánh: *<, <=, >, >=, =, <>.*

- Biểu thức trong cặp dấu ngoặc *( )* được thực hiện trước tiên nếu có.
- Các toán tử cùng thứ tự ưu tiên thì được thực hiện từ trái qua phải.

4 Ví dụ việc sử dụng các toán tử và toán hạng:

*3 + 5 * 3 = 18*

*(3 + 5) * 3 = 24*

*5 / 2 * 3 = 7.5*

(5 + 2 > 4) and not (true or (5 - 3 = 8)) = false

*(-b + sqrt(d)) / 2 * a* (có nghĩa: $\frac{-b+\sqrt{d}}{2}a$)

IV. Câu lệnh đơn giản:

Sau phần khai báo dữ liệu là phần lệnh của chương trình. Phần này xác định các công việc mà chương trình phải thực hiện xử lý các dữ liệu đã được khai báo. Câu lệnh được chia thành hai loại:

- Câu lệnh đơn giản:
  + Lệnh gán *(:=)*
  + Lệnh Nhập - Xuất *(READ, READLN, WRITE, WRITELN).*
  + Gọi thủ tục.
  + Lệnh nhảy *(GOTO).*
- Câu lệnh có cấu trúc:
  + Lệnh ghép *(BEGIN... END)*
  + Lệnh lựa chọn *(IF... ELSE, CASE... OF)*
  + Lệnh lặp *(FOR, REPEAT... UNTIL, WHILE... DO)*
  + Lệnh *WITH.*

Ö Ghi chú: Nội dung bài này chỉ đề cập đến các lệnh đơn giản. *Các lệnh có cấu trúc được trình bày ở bài 4.*

1. Lệnh gán:

Lệnh gán dùng để gán giá trị của một biểu thức *(có thể là hàm, biến hoặc giá trị)* cho một biến.

Cú pháp:

Biến := biểu_thức;

F Đầu tiên, *máy tính giá trị của biểu thức ở vế phải,* sau đó*, giá trị tính được từ vế phải được gán cho vế trái (biến).*

Ö Chú ý:

- *Vế trái của lệnh gán chỉ có thể là biến.* Ví dụ: viết *x + y = 7;* là sai vì vế trái của câu lệnh này là một *biểu thức chứ không phải là một biến.*
- Kiểu giá trị của biểu thức *(hàm, biến hoặc giá trị)* ở vế phải *phải trùng với kiểu của biến đã được khai báo*, trừ một số trường hợp như biến *kiểu thực (Single, Real, Double)* có thể nhận giá trị *kiểu nguyên (Shorint, Byte, Integer, Word, Longint),...* do tập hợp số nguyên là tập con của số thực.

4 Ví dụ: Sau khi đã có khai báo:

```
Var       c1, c2 : Char;
  i, j           : Integer;
  x, y           : Real;
```

thì ta có thể thực hiện các phép gán sau:

```
c1 := 'A';
c2 := Chr(97);
i  := (23 + 6) * 2 mod 3;
j  := Round(20 / 3);
x  := i;
y  := j;
```

2. Lệnh Xuất:

Lệnh xuất dùng để in lên màn hình các dữ liệu, kết quả hay các thông báo.

Cú pháp *(1). WRITE(Biểu_thức_1, Biểu_thức_2,..., Biểu_thức_n);*

(2). WRITELN(Biểu_thức_1, Biểu_thức_2,..., Biểu_thức_n);

(3). WRITELN;

Dạng *(1)*: In lên màn hình giá trị các biểu thức tại vị trí hiện hành của con trỏ theo thứ tự viết trong lệnh. Sau khi thực hiện xong lệnh *WRITE(...);* con trỏ định vị tại sau giá trị *biểu_thức_n* của câu lệnh.

Dạng *(2)*: In lên màn hình giá trị các biểu thức tại vị trí hiện hành của con trỏ theo thứ tự viết trong lệnh. Sau khi thực hiện xong lệnh *WRITELN(...);* con trỏ định vị tại đầu dòng kế tiếp.

Dạng *(3)*: Dùng để chuyển con trỏ xuống dòng.

4 Ví dụ:

```
Var  a, b   : Byte;
Begin
   A := 2;
   B := 4;
   Write   ( ' Day la ket qua phep nhan A voi B: ', a * b);
   Writeln;
   Writeln( '                    * * * *                    ');
   Write   ( ' ---------------------------------------------- ');
End.
```

1 Kết quả sau khi chạy chương trình trên:

```
Day la ket qua phep nhan A voi B: 8
                * * * *

------------------------------------------
```

Ö Chú ý: Có hai dạng viết trong thủ tục *Write* và *Writeln* là *viết không quy cách và viết có quy cách.* Điều này ta xét qua từng kiểu dữ liệu.

(1). Ví dụ về các dạng viết không có quy cách:

```
Uses  CRT;
Var
   I    : Integer;   R : Real;
   Ch  : Char;
   B    : Boolean;
Begin
   I := 123;  R := 123.456;  Ch := 'A';  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.
```

F *Cách viết không quy cách sẽ canh nội dung theo lề bên trái.*

- Số nguyên được viết ra với số chỗ đúng bằng số chữ số gán vào, kể từ vị trí bên trái. Lệnh {1} in ra: *123*

- Số thực được viết ra với trình tự sau: *một dấu cách*, tiếp đến là *một số phần nguyên, dấu chấm, 10 vị trí số thập phân,* tiếp đến là chữ *E*, *dấu của phần mũ (+,-), hai số biểu diễn giá trị phần mũ:*

  + Lệnh {2}in ra: *1.2345600000E+02*

  + Lệnh {3}in ra: *3.1400000000E+00*

  + Lệnh {4}in ra: *5.0000000000E+01*

- Kiểu ký tự in bình thường, một ký tự chiếm một chỗ. Lệnh {5}in ra: *A*

- Kiểu *Boolean* in ra một trong hai từ *True* hoặc *False*. Lệnh {6}in ra: *True*

- Lệnh {7}: *phát ra một tiếng Beep ổ loa.*

(2). Ví dụ về các dạng viết có quy cách:

```
Var
   I : Integer;
   R , Z : Real;
   Ch : Char;
   B : Boolean;
Begin
   I := 123;  R := 123.456;  Ch := 'A';  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.
```

F *Cách viết có quy cách sẽ canh nội dung theo lề bên phải, nếu thừa chỗ thì phần lề bên trái được để trắng.*

- Lệnh {1} và {2} dành 8 ký tự trên màn hình để in các số nguyên.

- Lệnh {3} và {4} dành 12 ký tự trên màn hình để in các số thực với 6 số lẻ phần thập phân, kết quả in ra: *123.456000* và *35.123457 (do phần thập phân >6 chỗ nên được làm tròn số).*

- Lệnh {5}in giá trị của *R* với *12 chỗ dạng mũ số: 1.23456E+02*

- Lệnh {6},{7} dành 5 chỗ để in chữ *A* và xâu ký tự *ABC.*

- Lệnh {8} dành 7 ký tự để in giá trị *True.*

- Lệnh {9} in số thực *Z* như sau: *Writeln(Z : m : n).* Nếu *m < n* thì số thực *Z* được in với *n* số lẻ, còn số chỗ trên màn hình thì tuỳ vào độ dài của số *Z.* Trong trường hợp *m > n* và độ dài của số lớn hơn *m* thì số được tự động canh phải. Trường hợp *m > n* và độ dài của số nhỏ hơn *m* thì số được canh phải dư bao nhiêu ký tự máy để trống bên trái.

Ö Trường hợp trong câu cần hiển thị dấu ' thì ta phải viết hai dấu ' liền nhau ('').

4 Ví dụ: *Write('Don''t forget me ! ');*

1 Kết quả: Trên màn hình hiển thị:

Don't forget me !

Ö Ghi chú: Muốn in dữ liệu ra máy in ta dùng lệnh *Write* hoặc *Writeln* với tham số *LST* vào trước. Biến *LST* được khai báo trong *Unit Printer*, vì vậy, để sử dụng lệnh in ta cần phải khai báo thư viện *Printer* trong chương trình.

4 Ví dụ:

```
Uses  Printer;
```

```
Begin
   Writeln(Lst,' Welcome to Turbo Pascal Language ! ');
End.
```

1 Kết quả: Khi chạy máy in ra giấy câu *Welcome to Turbo Pascal Language !*

3. Lệnh Nhập:

Lệnh nhập dùng để đưa dữ liệu từ bàn phím vào các biến.

Cú pháp:

(1) *Readln(Biến_1, biến_2, biến_n);*
(2) *Read(Biến_1, biến_2, biến_n);*

Khi thực hiện lệnh này, máy dừng lại chờ người dùng nhập vào đủ *n* lần nhập dữ liệu tương ứng với *n* biến.

Ngoài ra, ta có thể sử dụng thủ tục *Readln* để dừng chương trình và chờ người dùng ấn một phím bất kỳ để tiếp tục, ký tự được ấn không hiển thị lên màn hình.

Ö **Chú ý:**

- Các biến trong thủ tục *Readln* phải thuộc kiểu *nguyên, thực, ký tự* hoặc *xâu ký tự*. Do đó, ta không thể nạp từ bàn phím giá trị *True* hoặc *False* các biến kiểu *Boolean*.
- Dữ liệu nhập vào phải tương ứng với kiểu đã khai báo. Phải ấn phím *Enter* để thực hiện lệnh nhập sau khi gõ xong giá trị cần nhập.

4 **Ví dụ 1:** Với *a, b* là hai biến nguyên, *x* là biến thực. Xét đoạn chương trình sau:

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

Nếu ta gõ các phím: *2 24 6.5 14 <Enter>*

1 Kết quả: *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 chờ nhập số liệu cho biến *x*.

4 **Ví dụ 2:** Giả sử ta đã khai báo: *Var s1, s2, s3 : String[5];*

Xét câu lệnh: *Readln(s1, s2, s3);*

Nếu ta không nhập ký tự mà chỉ ấn *<Enter>* thì cả 3 biến *s1, s2, s3* đều là xâu rỗng.

Nếu ta gõ *ABCDE1234567* và ấn phím *< Enter >* thì: *s1 = 'ABCDE', s2 = '12345', s3 = '67'.*

4 Ví dụ 3: Viết chương trình tính diện tích *S* của hình thang với đáy dài *a*, đáy ngắn *b*, chiều cao *h*, tất cả được nhập từ bàn phím.

```
Program  DienTichHinhThang;
Uses  CRT;
Var  a, b, h, s : Real;
Begin
   ClrScr;
   Write( ' Nhap gia tri cua a, b, h :' );
   Readln(a, b, h);
   S := (a + b) * h / 2;
   Write( ' Dien tich S =  ',S:1:5);
   Readln;
End.
```

1 Kết quả khi chạy chương trình:

Nhap gia tri cua a, b, h : 5 3 4 < Enter >
Dien tich S = 16.00000

Ö Chú ý: Với cách lấy 3 giá trị bằng một lệnh *Readln(a, b, c);* thì các giá trị ta cần nhập cho mỗi biến phải cách với các giá trị khác ít nhất một ký tự trắng. Ta có thể nhập *a, b, c* bằng 3 lệnh *Readln(a); Readln(b); Readln(c);*

_____ o ² o _____

BÀI 4: CÁC LỆNH CÓ CẤU TRÚC TRONG NGÔN NGỮ PASCAL

I. Lệnh ghép:

Lệnh ghép là một nhóm các câu lệnh được đặt giữa hai từ khoá *BEGIN* và *END*. Lệnh ghép được thực hiện bằng cách thực hiện tuần tự các câu lệnh nằm giữa *BEGIN* và *END.*

Cú pháp:

```
Begin
   <câu lệnh 1>;
   <câu lệnh 2>;
```

```
    ...
  <câu lệnh n>;
End;
```

Sau *<câu lệnh n>* có thể có dấu ';' hoặc không. *Lệnh ghép cũng là một dạng câu lệnh.*

4 Ví dụ:

```
Begin
  temp := x;
  x := y;
  y := temp;
End;
```

Ö Chú ý: Sau từ khóa *END* có thể có dấu ';' hay không tùy thuộc vào các lệnh cấu trúc kế tiếp ta được học.

II. Lệnh lựa chọn:

1. Lệnh IF:

Cú pháp:

```
IF <biểu thức logic> THEN
   <lệnh 1>
ELSE
   <lệnh 2>;
```

Lệnh IF có thể không có phần *ELSE <lệnh 2>.*

F *Giải thích lệnh:* Khi gặp lệnh này máy kiểm tra *<biểu thức logic>,* nếu biểu thức này có giá trị *TRUE (tức là đúng như điều kiện đặt ra)* thì máy thực hiện *<lệnh 1>* nếu ngược lại, tức *<biểu thức logic>* có giá trị *FALSE* thì *<lệnh 2>* được thực hiện. Trường hợp trong câu lệnh không có phần *ELSE* và *<biểu thức logic>* có giá trị *FALSE* thì *<lệnh 1>* không được thực hiện và máy chuyển đến câu lệnh kế sau lệnh *IF* đó.

Ö Chú ý: câu lệnh trước từ khóa *ELSE* không được có dấu ';'. Trường hợp có câu lệnh ghép được đặt kế trước *ELSE* thì từ khoá *END* trước *ELSE* không được đặt dấu ';'.

4 Ví dụ 1: Chương trình nhập từ bàn phím 2 số nguyên *a, b.* Kiểm tra và cho biết số nào lớn hơn.

```
Var  a, b : Integer;
Begin
   Write( ' Nhap so a: ' );
   Readln(a);
   Write( ' Nhap so b: ' );
   Readln(b);
   If a > b then
      Write( ' So lon hon la ', a)  { tại vị trí này không được đặt dấu; }
   Else
      Write( ' So lon hon la ', b);
   Readln; { có thể không có dấu; tại câu lệnh cuối này }
End.
```

4 Ví dụ 2: Viết chương trình kiểm tra trong ba số *a, b, c* được nhập từ bàn phím, số nào là lớn nhất.

```
Var  a, b, c, max : Integer;
Begin
   Write( ' Nhap so a: ' );
   Readln(a);
   Write( ' Nhap so b: ' );
   Readln(b);
   Write( ' Nhap so c: ' );
   Readln(c);
   Max := a;
   If max < b then
      Max := b;
   If max < c then
      Max := c;
      Write( ' So lon hon la ', max);
   Readln;
End.
```

4 Ví dụ 3: Viết chương trình kiểm tra ba số được nhập từ bàn phím có thể là độ dài của ba cạnh trong một tam giác hay không? Nếu đúng là ba cạnh của tam giác thì

tính chu vi và diện tích tam giác, xét tam giác có phải là tam giác *đều*, *cân* hay không.

```
Var   a, b, c, p, s : Real;
   Begin
      Write( ' Nhap ba so a, b, c : ' );
      Readln(a, b, c);
      If (a>0) and (b>0) and (c>0) and (a+b>c) and (a+c>b) and (b+c>a) then
         Begin
            Writeln( ' Ba canh tren tao thanh mot tam giac. ' );
            If (a=b) and (b=c) then write( ' Day la tam giac deu. ' );
            If (a=b) or (a=c) or (b=c) then write( ' Day la tam giac can. ' );
            p := (a + b + c) / 2;
            s := SQRT(p * ( p - a ) * ( p - b ) * ( p - c ) );
            Writeln( ' Chu vi: ',2 * p:0:5, '. Dien tich:', s:0:5);
         End
      Else
         Write( 'Ba so nay khong tao thanh duoc mot tam giac.' );
   Readln;
   End.
```

2. Lệnh CASE:

Câu lệnh *IF* ở trên chỉ rẽ vào một trong hai nhánh tương ứng với giá trị của biểu thức logic. Còn lệnh *CASE (rẽ nhánh theo giá trị)* cho phép lựa chọn để thực hiện một trong nhiều công việc tùy theo giá trị của biểu thức.

Cú pháp:

CASE <biểu thức> OF
Tập_hằng_1: <lệnh_1>;
Tập_hằng_2: <lệnh_2>;
.....
Tập_hằng_n: <lệnh n>;
ELSE
<lệnh n +1>;
END;

Lệnh *CASE* có thể không có phần *ELSE <lệnh n +1>;*

F *Giải thích lệnh:*

1. *Tập_hằng_i (i = 1,..., n)* có thể bao gồm các *hằng* và các *đoạn hằng*, ví dụ:

3 : *<lệnh 1>;*
5, 10.. 15 : *<lệnh 2>;*
'A', Chr(152) : *<lệnh 3>;*
'0'.. '9' : *<lệnh 4>;*

2. Giá trị của *<biểu thức>* và giá trị trong các *Tập_hằng_i* phải có cùng kiểu và phải là *kiểu vô hướng đếm được (như nguyên, logic, ký tự, liệt kê).*

3. Tập hằng nào có chứa giá trị tương đương với giá trị của *<biểu thức>* thì lệnh sau dấu ':' của tập hằng đó được thực hiện, sau đó máy thoát khỏi lệnh *CASE.*

4. Trong trường hợp tất cả các tập hằng không có chứa giá trị tương đương với giá trị của *<biểu thức>* thì lệnh sau từ khóa *ELSE* được thực hiện. Trường hợp này nếu không có cả phần *ELSE <lệnh n+1>;* thì lệnh *CASE* này được thoát và không có lệnh nào sau dấu ':' được thực hiện.

4 Ví dụ 1: Viết chương trình nhập vào một điểm kiểm tra từ bàn phím và in kết quả xếp loại: loại *Yếu (dưới 5 điểm),* loại *Trung bình (5, 6 điểm),* loại *Khá (7, 8 điểm),* loại *Giỏi (9, 10 điểm).*

```
Var  Diem : Byte;
Begin
   Write( ' Nhap diem : ');
   Readln(Diem);
   Case  Diem  of
      0.. 4 : Write( ' Xep loai yeu. ' );
      5.. 6 : Write( ' Xep loai Trung binh. ' );
      7.. 8 : Write( ' Xep loai Kha. ' );
      9..10: Write( ' Xep loai Gioi. ' );
   Else
      Write( ' Diem nhap sai. ' );
   End;
   Readln;
End.
```

4 Ví dụ 2: Viết chương trình cho biết số ngày của một tháng. Thuật toán như sau:

- Nhập tháng vào biến *Thang*.
- Sau đó, dựa vào biến *Thang* để biết số ngày, số ngày này được đưa vào biến *SoNgay*. Trường hợp:
  + Tháng 1, 3, 5, 7, 8, 10, 12: *SoNgay := 31;*
  + Tháng 2:
    - Yêu cầu nhập năm vào biến *Nam*.
      - Ø Trường hợp *Nam* chia hết cho 4: *SoNgay := 29;*
      - Ø Trường hợp *Nam* không chia hết cho 4: ***SoNgay*** *:= 28;*
  + Tháng 4, 6, 9, 11: *SoNgay := 30;*
- In nội dung biến *SoNgay*.

```
Uses  CRT;
Var  SoNgay, Thang : Byte;
Nam : Integer;
Begin
   ClrScr;
   Write( ' Ban kiem tra thang may (dang so): ' );
   Readln(Thang);
   Case  Thang  of
      4, 6, 9, 11 :          SoNgay := 30;
      2 :                    Begin
                                Write( ' Thang nay thuoc nam nao (4 chu so): ' );
                                Readln(Nam);
                                If  Nam  mod 4 = 0 then  SoNgay := 29;
                                Else  SoNgay := 28;
                             End
   Else
      SoNgay := 31;
   End;
   If  Thang = 2 then
      Writeln( ' Thang ', thang , ' / ' , nam , ' co ' , SoNgay , ' ngay. ' )
   Else Writeln( ' Thang ', thang , ' co ' , SoNgay , ' ngay. ' );
   Readln
End.
```

III. Các câu lệnh lặp:

Trường hợp để giải quyết bài toán nào đó mà ta cần phải lặp đi lặp lại một công việc nào đó thì ta sẽ cần đến lệnh lặp. Số bước lặp có thể xác định hoặc không xác định. Trong ngôn ngữ Pascal có ba câu lệnh lặp là *FOR, REPEAT, WHILE.* Nếu số vòng lặp xác định thì ta sử dụng lệnh FOR còn vòng lặp không xác định thì ta sử dụng lệnh *REPEAT* hoặc *WHILE.* Tất cả các loại lệnh lặp phải có điểm dừng, cho dù đó là loại xác định hay không xác định.

1. Câu lệnh FOR:

Vòng lặp *FOR* có hai dạng là dạng *vòng lặp tiến* và *vòng lặp lùi.*

a. Dạng tiến:

Cú pháp: *FOR Biến := Biểu_thức1 TO Biểu_thức2 DO < Lệnh >*

Biến trong cấu trúc *FOR* gọi là biến điều khiển. Kiểu của biến điều khiển, *Biểu_thức1, Biểu_thức2* phải là k*iểu vô hướng đếm được (như nguyên, logic, ký tự, liệt kê).*

F *Giải thích sự hoạt động lệnh FOR dạng tiến:*

(1). Đầu tiên, *Biến* nhận giá trị của *biểu_thức1.*

(2). Máy kiểm tra *Biến* có nhỏ hơn hoặc bằng *biểu_thức2* hay không tức là xét điều kiện *(Biến <= Biểu_thức2) ?*

(3). Nếu điều kiện trên là sai thì máy thoát khỏi vòng lặp *FOR* để thực hiện các lệnh kế tiếp sau vòng lặp *FOR.* Nếu điều kiện trên là đúng thì *<Lệnh>* được thực hiện, sau đó, *Biến* được tăng một giá trị và quay trở lại bước (2).

<Lệnh> sẽ được thực hiện *((biểu_thức2 - biểu_thức1) + 1)* lần.

b. Dạng lùi:

Cú pháp: *FOR Biến := Biểu_thức1 DOWNTO Biểu_thức2 DO <Lệnh>*

F *Giải thích sự hoạt động lệnh FOR dạng lùi:*

(1). Đầu tiên, *Biến* nhận giá trị của *biểu_thức1.*

(2). Máy kiểm tra *Biến* có lớn hơn hoặc bằng *biểu_thức2* hay không tức là xét điều kiện *(Biến >= Biểu_thức2) ?*

(3). Nếu điều kiện trên là sai thì máy thoát khỏi vòng lặp *FOR* để thực hiện các lệnh kế tiếp sau vòng lặp *FOR.* Nếu điều kiện trên là đúng thì *<Lệnh>* được thực hiện, sau đó, *Biến được giảm một giá trị và quay trở lại bước (2).*

Ö Chú ý:

- Không được thay đổi giá trị của biến điều khiển bằng một lệnh bất kỳ trong vòng lặp *FOR*. Điều này có thể làm cho vòng lặp không có lối thoát và *dẫn đến treo máy.*

- Các *Biểu_thức1* và *Biểu_thức2 được ước lượng trước khi vào vòng lặp,* do đó số vòng lặp không bị thay đổi. Ta có thể lợi dụng tính tăng hoặc giảm của biến điều khiển để gán giá trị của nó cho bất kỳ biến nào hoặc thực hiện công việc nào đó có tính chất tăng hoặc giảm.

4 Ví dụ 1: Chương trình in lên màn hình 3 câu *Chào các bạn !* có số thứ tự đứng trước mỗi câu.

```
Uses  CRT;
Var  I : integer;
Begin
   ClrScr;
   For  I := 1  to  5  do
      Writeln( I , ' => ', ' Chao cac ban ' );
   Readln;
End;
```

4 Ví dụ 2: In lên màn hình 4 dòng chữ cái in thường và IN HOA theo chiều xuôi và chiều ngược.

```
Uses  CRT;
Var  kt : Char;
Begin
   ClrScr;
   For  kt := 'a'  to  'z'  do
      Write(kt : 3);
   Writeln;
   For  kt := 'z'  Downto  'a'  do
      Write(kt : 3);
   Writeln;
   For  kt := 'A'  to  'Z'  do
      Write(kt : 3);
   Writeln;
   For  kt := 'Z'  Downto  'A'  do
```

```
        Write(kt : 3);
    Readln;
End.
```

4 Ví dụ 3: Chương trình in lên màn hình 256 ký tự của bảng mã *ASCII.*

```
Var  i : Byte;
Begin
   For  i := 0  to  255  do
      Begin
         Writeln( ' Ma thu ' , i , ' la : ' , CHR(i) );
         If  (i+1)  mod 22 =  0  then
            Begin
               Write( ' An phim bat ky de xem tiep ! ' );
               Readln;
            End;
      End;
Readln;
End.
```

2. Câu lệnh Repeat:

Cú pháp:

REPEAT
<Lệnh 1>;
<Lệnh 2>;
.....
<Lệnh n>;
UNTIL < Biểu thức logic >;

F *Giải thích sự hoạt động lệnh REPEAT:*

Đầu tiên, thực hiện lần lượt các lệnh *<Lệnh 1>, <Lệnh 2>,..., <Lệnh n>,* sau đó kiểm tra *< Biểu thức logic >*. Nếu *< Biểu thức logic >* nhận giá trị *FALSE* thì lại quay lên đầu vòng lặp thực hiện tiếp *<Lệnh 1>, <Lệnh 2>,..., <Lệnh n>*. Nếu *< Biểu thức logic >* nhận giá trị *TRUE* thì máy thoát khỏi vòng lặp. *Như vậy, các lệnh nằm giữa REPEAT... UNTIL* được thực hiện ít nhất một lần.

Ö Chú ý:

- Các lệnh nằm giữa *REPEAT* và *UNTIL* không có từ khoá *Begin* và *End.*

- Trong vòng lặp phải có lệnh nào đó làm thay đổi giá trị một biến trong *<Biểu thức logic>* nhằm làm dừng vòng lặp, nếu không vòng lặp sẽ chạy mãi không ngừng dẫn đến treo máy.

4 Ví dụ 1: Chương trình yêu cầu nhập vào một mật khẩu là *'ttthcn'* thì mới thoát khỏi chương trình.

```
Uses  CRT;
Var  Password : String[6];
Begin
   Repeat
      Write( ' Xin hay nhap mat khau : ' );
      Readln(Password);
   Until Password = 'ttthcn';
   Write( ' Ban da nhap dung mat khau ! ' );
   Delay(1000);
   Readln;
End.
```

F *Giải thích lệnh*: *Delay(1000):* Thủ tục *Delay(n)* là thủ tục của *Unit CRT* tức là dừng một khoản thời gian là *1000 xung nhịp* của máy, vì vậy, tùy theo tốc độ của máy mà có khoản thời gian thực dừng lại khác nhau.

4 Ví dụ 2: Chương trình để sử dụng bàn phím giả thành phím đàn Piano với quy định: ấn phím *D phát ra nốt Do, phím R là nốt Re, M = Mi, F = Fa, S = Sol, L = La, S = Si.*

```
Uses  CRT;
Var  node : Char;
Begin
   ClrScr;
   Writeln( ' D = Do | R = Re | M = Mi | F = Fa | S = Sol | L = La | X = Si ' );
   Writeln( ' Q = Do cao | W = Re cao | E = Mi cao | K = Ket thuc ' );
   Repeat
      Node := ReadKey;
      Case  Node  of
         'd'   : Begin  NoSound;  Sound(262);  End;
         'r'   : Begin  NoSound;  Sound(294);  End;
         'm'  : Begin  NoSound;  Sound(330);  End;
```

```
            'f'   : Begin  NoSound;  Sound(349);  End;
            's'   : Begin  NoSound;  Sound(392);  End;
            'l'   : Begin  NoSound;  Sound(440);  End;
            'x'   : Begin  NoSound;  Sound(494);  End;
            'q'   : Begin  NoSound;  Sound(523);  End;
            'w'   : Begin  NoSound;  Sound(587);  End;
            'e'   : Begin  NoSound;  Sound(659);  End;
         End;
      Until  (Upcase(Node) = 'K');
      NoSound;
   End.
```

Ö **Ghi chú:** Thủ tục *Sound(n)* dùng để phát một âm thanh có tần số *n Hertz* cho đến khi gặp hàm *NoSound (ngừng phát âm thanh),* hai thủ tục trên thường đi đôi với nhau khi sử dụng. Những chương trình cần sự lặp đi lặp lại theo ý muốn thường sử dụng vòng lặp *Repeat... Until.* Cách thực hiện như sau:

```
Var  TiepTuc : Char;
.....
Begin
   Repeat
      <... Các lệnh của chương trình >
      Write( ' Co tiep tuc nua khong (C/K) ? ' );
      Readln(TiepTuc);
   Until  Upcase(TiepTuc) = 'K';
End.
```

3. Câu lệnh While:

Cú pháp:

WHILE < Biểu thức logic > DO
< Lệnh >;

F *Giải thích lệnh:* Gặp lệnh này trước tiên máy kiểm tra *< Biểu thức logic >,* nếu nó có giá trị *TRUE* thì thực hiện *< Lệnh >* và sau đó quay lại kiểm tra *< Biểu thức logic >* và quá trình cứ tiếp tục như vậy. Nếu *< Biểu thức logic >* nhận giá trị *FALSE* thì máy lập tức thoát khỏi vòng lặp. Như vậy lệnh *WHILE* dùng để lặp đi lặp lại một công việc trong khi điều kiện còn được thỏa mãn.

Ö **Ghi chú:** Nếu ngay từ khi mới vào vòng lặp mà thấy điều kiện không được thỏa mãn, máy tự động thoát ngay mà không thực hiện *<Lệnh>* bên trong vòng lặp.

4 Ví dụ: Chương trình tìm ước số chung lớn nhất của hai số nguyên.

```
Var  a, b, r : Integer;  tl : Char;
Begin
   Repeat
      Write( ' Nhap hai so a va b : ' );
      Readln(a, b);
      While  b <> 0  do
         Begin
            r := a mod b;
            a := b;
            b := r;
         End;
      Writeln( ' Uoc so chung lon nhat la ' , a );
      Write( ' Ban tim USCLN nua khong (C/K) ? );
      Readln(tl);
   Until Upcase(tl) = 'K';
End.
```

IV. Các lệnh Goto, Break, Exit và Halt:

1. Lệnh Goto:

Cú pháp:

GOTO Lab;

Trong đó, *Lab* là một nhãn. Nhãn là một tên *như tên biến* hoặc là *một số nguyên từ 0 đến 9999.* Tên nhãn được khai báo theo hướng dẫn ở bài1 (IV.2).

Khi gặp lệnh *Goto Lab,* máy nhảy không điều kiện đến thực hiện câu lệnh sau nhãn *Lab.*

Lệnh Goto chỉ cho phép nhảy từ vị trí này đến vị trí khác trong cùng một thân hàm, thủ tục, cho phép nhảy từ trong một vòng lặp ra ngoài; không cho phép nhảy từ ngoài vào trong một vòng lặp, thủ tục, hàm hoặc khối lệnh.

4 Ví dụ: Chương trình tìm các số nguyên tố nằm giữa hai số nguyên dương *n1* và *n2*, hai số này được nhập từ bàn phím *(khái niệm số nguyên tố: là số nguyên chỉ chia chẵn hết cho 1 và chính nó).*

```
Program  NguyenToByGoto;
Label  L1, L2;
Var   i, j, n1, n2 : Integer;
      TL : Char;
Begin
   L1: Write( ' Nhap hai gia tri nguyen : ' );
   Readln(n1, n2);
   For  i := n1  to  n2  do
      Begin
         For j := 2  to  i - 1 do
            If (i mod j = 0)  then  Goto L2;
         Write( i, '   ' );
         L2: ; {; cũng là một lệnh, nhưng là lệnh rỗng, tức là không làm gì cả }
      End;
   Writeln;
   Write( ' Ban muon tiep tuc khong ? (C/K) ' );
   Readln(TL);
   If  (Upcase(TL) = 'C')  then  Goto  L1;
End.
```

2. Lệnh Break:

Trong thân các lệnh lặp *FOR, WHILE, REPEAT* khi gặp lệnh *Break* thì máy sẽ thoát khỏi chu trình. Nếu có nhiều lệnh lặp lồng nhau thì máy thoát khỏi chu trình trong nhất chứa lệnh *Break.*

4 Ví dụ: In ra màn hình 4 dãy số từ *1* đến *49.*

```
Uses  CRT;
Var  i, j : Integer;
Begin
   ClrScr;
   For  j := 1  to  4  do
      Begin
         Writeln;
         Writeln( ' j = ' , j );
         For  i := 1  to  300  do  { * }
            Begin  If  i = 50  then
```

```
                Break; { Thoát khỏi vòng lặp For * }
                Write( i, '  ' );
            End;
        Readln;
    End;
Readln;
End.
```

3. Lệnh Exit:

Nếu lệnh *Exit* thuộc chương trình con thì việc thực hiện *Exit* làm chấm dứt chương trình con, trở về chỗ gọi nó. Nếu lệnh Exit thuộc chương trình chính thì việc thực hiện nó sẽ làm chấm dứt chương trình.

4 Ví dụ: Chương trình cứ nhắc lại câu *Welcome to Turbo Pascal Language* sau mỗi lần ấn một phím. Chương trình sẽ thoát khi ấn phím *E* hoặc *e*.

```
Uses  CRT;
Label  L1;
Var  TL : Char;
Begin
   L1: Writeln( ' Welcome to Turbo Pascal Language ! ' );
   TL := Readkey; { Chờ một phím được ấn, giá trị được đặt vào biến TL, đây là
                    hàm của Unit CRT }
   If  (Upcase(TL) = 'E')  then
     Exit
   Else
     Goto  L1;
End.
```

4. Lệnh Halt:

Lệnh *Halt* dùng để dừng ngay chương trình đang chạy. Lệnh *Halt* thường được dùng khi phải một trường hợp nào đó mà thuật toán không thể tiếp tục được.

BÀI 5. DỮ LIỆU KIỂU VÔ HƯỚNG LIỆT KÊ VÀ KIỂU ĐOẠN CON

I. Kiểu liệt kê:

Kiểu liệt kê được định nghĩa bằng cách liệt kê tất cả các giá trị của kiểu thông qua các tên do người lập trình đặt ra và danh sách các giá trị trên được đặt trong cặp ngoặc đơn *( )*.

4 Ví dụ:

```
Type  Days = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
      Viec = (DiHoc, LamBai, ThiNghiem, Nghi);
```

Khi đó, ta có thể khai báo biến như sau:

```
Var  HomQua, HomNay : Days;
Lam : Viec;
```

Hoặc ta có thể khai báo trực tiếp với mô tả kiểu dữ liệu như sau:

```
Var  GioiTinh : (Nam, Nu);
Color : (Red, Blue, Green, White, Black);
```

Ö Chú ý:

(1). Có thể thực hiện phép gán trên các trị kiểu liệt kê, ví dụ:

```
Lam := Nghi;
Color := Blue;
```

(2). Các giá trị của các kiểu liệt kê có thể so sánh với nhau theo quy định: Giá trị đứng trước nhỏ hơn giá trị đứng sau. Ta chỉ sử dụng toán tử so sánh cho kiểu liệt kê và cũng là toán tử duy nhất dùng cho kiểu này.

4 Ví dụ: Theo như khai báo trên, nếu so sánh *Thu < Fri* cho kết quả *True*, hoặc *Red >= Blue* cho kết quả *False*.

(3). **Các hàm chuẩn áp cho kiểu liệt kê:**

- Hàm *ORD*: Cho thứ tự trị của đối số trong kiểu liệt kê.

4 Ví dụ: theo như khai báo trên, *ORD(Sun) = 0, ORD(Mon) = 1.*

- Hàm *PRED*: Cho trị đứng trước của đối số trong kiểu liệt kê.

4 Ví dụ: theo như khai báo trên, *PRED(Sat) = Fri, PRED(LamBai) = DiHoc. PRED(Sun)* **ð** lỗi chương trình.

- Hàm *SUCC*: Cho trị đi sau đối số trong kiểu liệt kê.

4 Ví dụ: theo như khai báo trên, *SUCC(Fri) = Sat. SUCC(Sat)* **ð** lỗi chương trình.

(4). Không thể *nhập, xuất đối với dữ liệu kiểu liệt kê*. Giá trị thuộc kiểu liệt kê thường được dùng để làm chỉ số cho vòng lặp *FOR,* các trường hợp lựa chọn trong lệnh *CASE*, chỉ số cho các mảng *(Array).*

4 Ví dụ: Chương trình đổi thứ trong tuần ra số. *Chủ nhật ứng với số 0, Thứ hai ứng với số 1,...*

```
Type
   Thu = (ChuNhat, ThuHai, ThuBa, ThuTu, ThuNam, ThuSau, ThuBay);
Var
   Ngay : Thu;
Begin
   Writeln( ' Chuong trinh doi thu ra so ' );
   For  Ngay := ChuNhat  to  ThuBay  do
      Write(Ord(Ngay));
   Readln;
End.
```

II. Kiểu đoạn con:

Kiểu đoạn con được định nghĩa do người dùng dựa trên cơ sở các kiểu vô hướng đếm được *(Nguyên, Logic, Ký tự, Liệt kê)* theo dạng:

Tên_kiểu_đoạn_con = Hằng_dưới.. Hằng_trên;

Trong đó: *Hằng_dưới, Hằng_trên* là các giá trị hằng có cùng kiểu giá trị và thoả mãn điều kiện: *Hằng_dưới < Hằng_trên*. Khi đó, các giá trị của kiểu đoạn con sẽ xác định trong khoản từ *Hằng_dưới* đến *Hằng_trên*.

4 Ví dụ:

```
Type
   Ky_so = '0'.. '9'; { Kiểu gồm các ký tự số từ  '0' đến '9'}
   Ngay = (Hai, Ba, Tu, Nam, Sau, Bay, ChuNhat);
   Ngay_Lam_Viec = Hai.. Bay; {Kiểu Ngay_Lam_Viec là khoản con của kiểu
Ngay }
```

ChiSo = 1.. 50; { Kiểu ChiSo gồm các số nguyên từ 1 đến 50 }
Tuoi_Lam_Viec = 18.. 50;

Kiểu miền con giúp cho chương trình dễ đọc, dễ kiểm tra và tiết kiệm bộ nhớ.

_____ o ² o _____

BÀI 6. KIỂU TẬP HỢP VÀ KIỂU MẢNG

I. Kiểu tập hợp:

1. Định nghĩa:

Dữ liệu kiểu tập hợp là một tập hợp của những dữ liệu cùng thuộc một kiểu vô hướng đếm được. Một kiểu tập hợp được khai báo theo dạng sau:

SET OF Kiểu_cơ_sở;

4 Ví dụ:

```
Type
   Chu_so = Set of 0.. 9;
   Chu_hoa = Set of 'A'.. 'Z';
Var
   So : Chu_so;
   Chu : Chu_hoa;
   Mau : Set of (Xanh, Vang, Tim);
```

Ö Chú ý:

- Các giá trị được đưa vào tập hợp cần có số thứ tự trong khoản từ 0 đến 255.
- Như vậy, với khai báo:

```
Type
   Tap_so = Set of 10.. 256;
```

1 Kết quả khi dịch máy sẽ thông báo lỗi: *Set base type out of range.*

- Một dữ liệu kiểu tập hợp có dạng các phần tử nằm trong hai dấu ngoặc *[ ].* Ví dụ: *['A', 'D', 'E'], [3,5..9];*
- Tập hợp rỗng ký hiệu là *[ ].*
- Biến tập hợp cho phép có từ *0 đến 256* phần tử.
- Có thể thực hiện phép gán trên kiểu tập hợp. Ví dụ:

```
So := [0, 4, 9];
Chu := [ ]; {Tập hợp rỗng}
```

Mau := [Vang, Tim];

2. Các phép toán trên tập hợp:

a. Phép toán quan hệ:

Phép toán = ð cho giá trị *True* nếu hai tập hợp bằng nhau.
Phép toán < > ð cho giá trị *True* nếu hai tập hợp khác nhau.
Phép toán <= ð *A <= B* cho giá trị *True* nếu *A* là tập con của *B.*
Phép toán >= ð *A >= B* cho giá trị *True* nếu *B* là tập con của *A.*

Ö **Chú ý:** Không có *phép toán* < và > cho kiểu tập hợp. Để kiểm tra tập hợp *A* có thật sự nằm trong *B* hay không ta dùng câu lệnh:

If (A<>B) and (A<=B) then Write('A la tap con that su cua B ');

b. Phép toán IN:

Phép toán *IN* dùng để xem xét một phần tử nào đó có nằm trong tập hợp không ? Nếu phần tử đó có trong tập hợp thì phép toán sẽ trả về giá trị *True*, ngược lại cho giá trị *False.* Ví dụ:

'C' In ['A', 'C', 'D'] cho kết quả *True.*
'E' In ['A', 'C', 'D'] cho kết quả *False.*

c. Phép toán hợp, giao, hiệu:

Gọi *A, B* là hai tập hợp cùng kiểu dữ liệu.
A + B là hợp của *A* và *B*: tập hợp các phần tử thuộc *A* hoặc thuộc *B.*
*A * B* là giao của *A* và *B*: tập hợp các phần tử thuộc *A* và thuộc *B.*
A - B là hiệu của *A* và *B*: tập hợp các phần tử thuộc *A* và không thuộc *B.*

4 Ví dụ:

```
A := [1, 3, 9];
B := [9, 2, 5];
```

Vậy:

*A * B* có giá trị là *[9].*
A - B có giá trị là *[1, 3].*

4 Ví dụ: Viết chương trình nhập vào một chữ cái. Xét xem chữ cái đó là nguyên âm hay phụ âm.

```
Var
  ChuCai, NguyenAm : Set of Char;
  Ch : char;
```

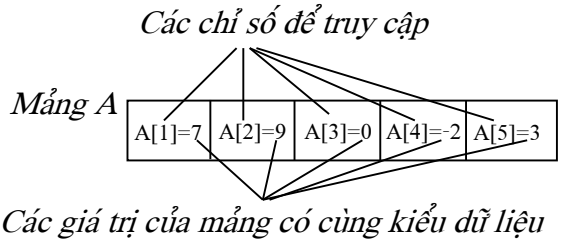
```
Begin
   ChuCai := ['A'.. 'Z', 'a'.. 'z'];
   NguyenAm := ['A', 'E', 'I', 'O', 'U'];
   Repeat
      Write( ' Nhap mot chu cai de kiem tra: ' );
      Readln(Ch);
   Until  Ch  IN  ChuCai;
   If  Upcase(Ch)  IN  NguyenAm  then
      Writeln(Ch, ' la nguyen am. ' )
   Else
      Writeln(Ch, ' la phu am. ');
   Readln;
End.
```

II. Kiểu mảng:

1. Khái niệm:

Mảng *(Array)* là một kiểu dữ liệu có cấu trúc bao gồm một số cố định các thành phần có cùng kiểu, có cùng một tên chung. Các thành phần của mảng được truy xuất thông qua các chỉ số.

4 Ví dụ: Mảng A gồm năm phần tử: *A[1]=7, A[2]=9, A[3]=0, A[4]= -2, A[5]=3:*



Công dụng của mảng là dùng để lưu trữ một dãy số liệu có cùng một tính chất nào đó. Ví dụ: các điểm kiểm tra một môn học nào đó của một học sinh, các giá trị của một dãy số được nhập từ bàn phím.

2. Khai báo mảng một chiều:

```
Type
    Tên_kiểu_mảng = ARRAY [Chỉ_số]  OF  Kiểu_phần_tử;
Var
    Tên_biến_mảng : Tên_ kiểu_mảng;
```

Trong đó:

- *Kiểu_phần_tử* là kiểu dữ liệu của mỗi phần tử trong mảng *(là kiểu bất kỳ).*
- *Chỉ_số* là danh sách các chỉ số để truy cập đến các thành phần của mảng.

Các chỉ số có thể là:

+ Một đoạn con, ví dụ:

```
Type
   Ho_Ten = Array[1..100]  of  String[30];
   He_so_luong = Array[1..100]  of  Real;
```

+ Một danh sách liệt kê, ví dụ:

```
Type
   Toc_do = Array[(Oto, Tai, Buyt, GanMay)]  of  Integer;
```

+ Một kiểu dữ liệu, ví dụ:

```
Type
   ASCIIType = Array[Byte]  of  Char;
   Xe = (Oto, Tai, Buyt, GanMay);
   Toc_do = Array[Xe]  of  Integer;
```

Với các kiểu mảng trên, ta có thể khai báo các biến mảng sau:

```
Var
  HeSo : He_so_luong;
  HT : Ho_Ten;
  Speed : Toc_do;
```

Ngoài cách định nghĩa *Tên_kiểu_mảng* như ở trên ta cũng có thể khai báo một biến mảng trực tiếp sau lệnh *VAR*:

```
Var  ch : Array[0.. 25]  of  Char;
     Th : Array[-2.. 4]  of  Real;
```

3. Truy cập các phần tử của mảng:

Việc truy nhập vào một phần tử nào đó của biến mảng được thực hiệnh qua tên biến mảng, theo sau là giá trị chỉ số đặt trong dấu *[ ].* Ví dụ:

```
Ch[2] := 'B';
Th[1] := 12.5;
HT[1] := 'Vu Duc Dung';
```

4 Ví dụ 1: Nhập *n* số thực từ bàn phím vào một mảng, tính trung bình cộng của các số này.

```
Uses CRT;
Var   i,n : Integer;
      s : Real;
      a : Array[1.. 100]  of  Real;
Begin
   ClrScr;
   Write( ' Ban muon nhap bao nhieu PT cho mang : ' );
   Readln(n);
   For  i := 1  to  n  do
      Begin
         Write( ' PT A[ ' , i , ' ]= ' );
         Readln(a[i]);
      End;
   s := 0;
   For  i := 1  to  n  do
      s := s + a[i];
   Write( ' Trung binh cong cua day so = ' , s / n : 0 : 4 );
   Readln;
End.
```

4 Ví dụ 2: Nhập từ bàn phím *n* phần tử thực của một mảng, sắp xếp dãy theo thứ tự tăng dần, xuất giá trị của mảng lên màn hình.

```
Var   a : array[1..10]  of  Real;
      b : array[1..10]  of  Real;
      temp : Real;
      i, j, n : integer;
Begin
   n:=10;
   For  i := 1  to  n  do
      Begin
         Write( ' PT thu ' , i , ':' );
         Readln( a[i] );
      End;
```

```
For i := 1 to n - 1 do
   For j := n downto i do
      If a[i] > a[j] then
         Begin
            temp := a[i];
            a[i]:=a[j];
            a[j]:=temp;
         End;
   For i := 1 to n do
   Write( a[i] : 0 : 3 , ' ' );
Readln;
End.
```

4. Mảng nhiều chiều:

Phần này chủ yếu trình bày các mảng hai chiều. Các mảng nhiều hơn hai chiều được suy diễn một cách tự nhiên.

Việc khai báo mảng hai chiều cũng giống như mảng một chiều, chỉ có điều khác là nó có hai tập chỉ số được viết cách nhau bởi dấu ','.

4 Ví dụ:

```
Type
   Mang1 = Array[1.. 30, 1.. 50] of Integer;
   Mang2 = Array[1.. 3, 0.. 2] of Real;
Var
   A : Mang1;
   B : Mang2;
```

Trong đó, số phần tử của mảng số thực *B* là *3 x 3 = 9 (phần tử),* sắp đặt trong bộ nhớ theo thứ tự như sau:

B[1, 0] B[1, 1] B[1 ,2]
B[2, 0] B[2, 1] B[2 ,2]
B[3, 0] B[3, 1] B[3 ,2]

Ö **Chú ý:** Mảng hai chiều còn gọi là ma trận. Trong ví dụ trên, *B* là ma trận cấp *3 x 3*. Trong mảng hai chiều, chỉ số sau truy cập nhanh hơn chỉ số trước. Để truy cập đến phần tử *hàng thứ i, cột thứ j* của mảng hai chiều *B* ta dùng cách viết:

B[i][j]

hoặc

B[i , j]

4 Ví dụ: Nhập một ma trận *m* hàng, *n* cột từ bàn phím. Tính và in ra màn hình tổng của mỗi cột và tổng của mỗi hàng.

```
Const  mMax = 30, nMax = 30;
Type
   Mang = Array[1.. mMax, 1.. nMax]  of  Real;
Var
   n, m, i, j : Integer;
   sum : Real;
   a : Mang;
Begin
Write( ' Ban muon nhap ma tran bao nhieu hang va cot ? ' );
Readln( m, n );
For  i := 1  to  m  do
   For  j := 1  to  n  do
      Begin
         Write( ' PT thu [ ' , i , ' , ' , j, ' ] = ' );
         Readln( a[ i, j ] );
      End;
For  j := 1  to  n  do
   Begin
      sum := 0;
            For  i := 1  to  m  do
                  Sum := sum + a[ i, j ];
            Write( ' Tong cot ' , j ,' = ' , sum : 0 : 5 );
    End;
For  i := 1  to  m  do
    Begin
            sum := 0;
            For  j := 1  to  n  do
                  Sum := sum + a[ i, j ];
            Write( ' Tong hang ' , i ,' = ' , sum : 0 : 5 );
    End;
Readln;
```

End.

_____ o ² o _____

BÀI 7. CHƯƠNG TRÌNH CON: HÀM VÀ THỦ TỤC

Khi lập trình, có những đoạn chương trình cần dùng nhiều lần. Để tránh việc viết lại đoạn này, ta nên chuyển đoạn chương trình này thành một chương trình con và mỗi lần cần thực hiện công việc đó thì ta gọi nó thông qua tên.

Chương trình con còn để mẫu hoá một chương trình làm công việc nào đó. Người khác dùng chương trình con *chỉ cần biết truyền số liệu vào và lấy kết qủa ra như thế nào mà không cần phải quan tâm đến thuật toán trong chương trình con như thế nào*.

Khi viết những chương trình lớn, để dễ dàng quản lý, gỡ rối và hiệu chỉnh chương trình, ta nên phân chương trình thành nhiều *công việc độc lập*, mỗi công việc là một chương trình con. Chương trình con gồm có hai loại là *HÀM (Function)* và *THỦ TỤC (Procedure).*

I. Hàm và thủ tục:

Cấu trúc của hàm có dạng:

```
FUNCTION  Tên_Hàm(ThamSố1: Kiểu; TS2: Kiểu;... ) : Kiểu;
Var  Các _biến_cục_bộ;
Begin
   Các lệnh tính toán;
      ...;
   Tên_Hàm := Giá_trị;
End;
```

Phương pháp gọi hàm: ta gọi hàm thông qua tên kèm theo tham số của hàm như sau:

Tên_hàm(Danh sách các tham số thực sự);

Cấu trúc của thủ tục có dạng:

PROCEDURE Tên_Thủ_tục(TS1: Kiểu; TS2: Kiểu;...; Var TS3: Kiểu; Var TS4: Kiểu;...);

```
Var  các biến cục bộ;
Begin
     Các lệnh;
     ...;
End;
```

Phương pháp gọi thủ tục:

Tên_hàm(Danh sách các tham số thực sự);

Sự *khác nhau cơ bản giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm, hàm có thể tham gia vào các biểu thức tính toán* còn *thủ tục không cho giá trị nào cả.* Khi tạo hàm, *trong thân hàm bao giờ cũng có giá trị gán cho tên hàm để hàm trả về giá trị này khi được gọi.*

Các tham số khác sau tên hàm và tên thủ tục gọi là các tham số hình thức *(hay còn gọi là đối). Trong thủ tục, các tham số hình thức có hai loại: các tham số được khai báo sau từ khoá Var gọi là tham số biến, các số khai báo không có từ khoá Var ở trước gọi là tham số giá trị. Trong hàm chỉ có tham số giá trị,* tức khai báo mà không có từ khoá *Var.*

Tham số thực sự là các tham số dùng trong lời gọi hàm hay thủ tục. Danh sách các tham số thực sự trong lời gọi hàm phải tương ứng với danh sách các tham số hình thức trong phần khai báo chương trình con và chúng phải tương ứng về kiểu.

Trong thủ tục, *các tham số giá trị thường là các biến để chứa dữ liệu đưa vào thủ tục;* các *tham số biến là các biến mà kết quả tính toán của thủ tục sẽ chứa vào đó khi ra khỏi thủ tục, ta có thể dùng chúng để tính toán tiếp.*

4 Ví dụ cách sử dụng tham số giá trị và tham số biến:

```
Var  a, b, c, d : Integer;
Procedure  Chuyen(x, y: Integr; Var u, v: Integer);
Begin { Từ khoá bắt đầu thủ tục Chuyen }
   x := 2 * x;
   y := 3 * y;
   u := 4 * u;
   v := 5 * v;
End;
Begin { Từ khoá bắt đầu chương trình chính }
   a := 10;
```

```
    b := 10;
    c := 10;
    d := 10;
    Chuyen(a, b, c, d);
    Write( ' a = ', a, '. b = ', b, '. c = ', c, '. d = ', d );
    Readln;
End.
```

1 Kết quả khi chạy chương trình: *a = 10. b = 10. c = 40. d =50*

II. Biến toàn cục, biến cục bộ và việc truyền dữ liệu:

Biến toàn cục là biến khai báo ở đầu chương trình chính, tồn tại trong suốt thời gian làm việc của chương trình. Ta có thể sử dụng và làm thay đổi giá trị của biến toàn cục nhờ các câu lệnh trong chương trình chính cũng như trong tất cả các chương trình con.

Biến cục bộ là biến là biến khai báo ở đầu chương trình con. Chúng được cấp phát bộ nhớ khi chương trình con đước gọi đến và bị xoá khi máy thoát khỏi chương trình con đó. Biến cục bộ có giá trị trong chương trình con và tất cả các chương trình con khác nằm trong chương trình con này.

Nếu tên biến cục bộ của một chương trình con trùng với một tên biến toàn cục thì máy không bị nhầm lẫn, máy sẽ dùng hai ô nhớ khác nhau để lưu trữ hai biến, khi ra khỏi chương trình con, biến cục bộ tự động được xoá.

Khi gặp một lời gọi đến chương trình con, máy sẽ thực hiện các bước sau:

- Cấp phát bộ nhớ cho các đối, các biến cục bộ.
- Truyền giá trị của các tham số thực sự cho các tham số giá trị tương ứng, truyền địa chỉ các tham số thực sự ứng với tham số biến cho các tham số biến của thủ tục.
- Thực hiện các lệnh trong chương trình con, trong khi thực hiện chương trình con, các biến cục bộ và các tham số giá trị có thể bị biến đổi nhưng không ảnh hưởng đến các biến bên ngoài. Trái lại, *mọi thay đổi của tham số biến trong chương trình con sẽ kéo theo sự thay đổi của tham số thực sự tương ứng (vì có sự truyền theo địa chỉ).* Do đó, *khi thoát khỏi chương trình con, các tham số thực sự ứng với tham số biến vẫn giữ được giá trị mới nhất do chương trình con tạo ra.*
- Thực hiện xong các lệnh của chương trình con, máy xoá tất cả các đối và các biến cục bộ và trở về lệnh kế sau nơi gọi nó.

Việc lấy kết quả thực hiện chương trình con như sau: nếu là *hàm* thì lấy kết quả thông qua tên hàm, nếu là *thủ tục* thì kết quả ở tham số thực sự ứng với tham số biến. Khi cần lấy duy nhất một giá trị từ chương trình con thì ta lập một *FUNCTION*, khi cần lấy từ hai giá trị trở lên từ chương trình con hoặc không lấy giá trị nào thì ta phải lập *PROCEDURE.*

4 Ví dụ 1: Lập hàm tính diện tích hình thang. Nhập dữ liệu của hai thửa ruộng hình thang và tính tổng diện tích hai thửa ruộng.

```
Var  a1, b1, h1, a2, b2 , h2, s : Real;
(************* Bat dau Function **************)
Function  DTHinhThang(a, b, h) : Real;
Begin
   DTHinhThang := (a + b) * h / 2;
End;
(********* Bat dau chuong trinh chinh **********)
Begin
   Write( ' Canh dai, ngan va cao cua thua ruong thu nhat: ' );
   Readln(a1, b1, h1);
   Write( ' Canh dai, ngan va cao cua thua ruong thu hai: ' );
   Readln(a2, b2, h2);
   s := DTHinhThang(a1, b1, h1) + DTHinhThang(a2, b2, h2);
   Writeln( ' Tong dien tich hai thua ruong = ', s : 0 : 3);
   Readln;
End.
```

4 Ví dụ 2: Lập hàm tính ước số chung lớn nhất *(USCLN).* Sau đó, dùng hàm này để tính *USCLN* và bội số chung nhỏ nhất *(BSCNN)* của hai số được nhập từ bàn phím.

```
Var m, n, usc, bsc: Integer;
(************  Function USCLN  *************)
Function USCLN(a, b : Integer): Integer;
Var  r : Integer;
Begin
   While  b < > 0  do
      Begin
         r := a  mod  b;
```

```
            a := b;
            b := r;
        End; { a hien tai la USCLN cua a va b ban dau  }
    USCLN := a;
End;
(*********  bat dau chuong trinh chinh  *********)
Begin
    Write( ' Nhap so thu nhat : ' );
    Readln(m);
    Write( ' Nhap so thu hai: ' );
    Readln(n);
    usc := USCLN(m, n);
    bsc := m * n div USCLN(m, n);
    Writeln( ' Uoc so chung lon nhat cua ', m, ' va ', n, ' la : ', usc);
    Writeln( ' Boi so chung nho nhat cua ', m, ' va ', n, ' la :', bsc);
    Readln;
End.
```

4 Ví dụ 3: Lập một thủ tục để tính đồng thời diện tích và thể tích hình cầu.

```
Var  r, s, v : Real;
Reply : Char;
(**************  Function  ***************)
Procedure  SVHinhCau( r : Real; Var  s, v :Real);
Begin
    s := 4 * pi * r * r;
    v := 4 * pi * r * r * r / 3;
End;
(********  bat dau chuong trinh chinh  ********)
Begin
    Repeat
        Write( ' Nhap ban kinh hinh cau : ' );
        Readln(r);
        SVHinhCau(r, s, v);
        Writeln( ' Dien tich = ', s : 0 : 4, '. The tich = ', v : 0 :4 );
        Write( ' Ban co tiep tuc khong ?(C/K) ' );
        Readln(Reply);
```

```
  Until Upcase(Reply) = 'K';
End.
```

III. Các hàm và thủ tục thường dùng của Unit CRT:

Unit CRT có nhiều hàm, thủ tục dùng để điều khiển màn hình, bàn phím và âm thanh. Nó cho phép mở các cửa sổ với các màu sắc khác nhau, thay đổi màu của các dòng chữ trên màn hình, giúp cho việc trình bày màn hình đẹp và hấp dẫn hơn, tổ chức hội thoại giữa người và máy thuận tiện. Khi dùng các hàm và thủ tục này, ở đầu chương trình chính cần phải có khai báo *USES CRT;* Các thủ tục của *Unit CRT* gồm:

1. Thủ tục ClrScr:

Xoá màn hình và đưa con trỏ về vị trí *(1,1)* trên màn hình. Màn hình mặc định được chia thành 25 dòng và 80 cột. Cột đầu tiên đánh số 1, dòng đầu tiên đánh số 1.

2. Thủ tục ClrEOL:

Xoá từ vị trí con trỏ đến cuối dòng hiện hành. Sau khi thực hiện xong, con trỏ đứng ngay vị trí trước khi gọi thực hiện thủ tục.

3. Thủ tục DelLine:

Xoá dòng con trỏ đang đứng, các dòng sau sẽ được chuyển lên trên một dòng.

4. Thủ tục InsLine:

Chèn dòng trống vào vị trí hiện hành của con trỏ trên màn hình.

5. Thủ tục GotoXY(x, y: Byte):

Đưa con trỏ đến, cột thứ x, dòng thứ y.

6. Hàm WhereX: Byte

Cho giá trị kiểu byte cho biết con trỏ đang ở cột nào.

7. Hàm WhereY: Byte

Cho giá trị kiểu byte cho biết con trỏ đang ở dòng nào.

8. Thủ tục Sound(Hz : Word):

Phát âm thanh có tần số Hz cho đến khi gặp thủ tục NoSound thì dừng lại.

9. Thủ tục NoSound:

Tắt loa phát âm thanh ở máy.

10. Thủ tục TextBackGround(Color : Byte):

Chọn màu nền trong chế độ văn bản *(Chế độ mặc định khi chạy Pascal).* Color có giá trị từ 0 đến 7.

11. Thủ tục TextColor(Color : Byte):

Chọn màu của ký tự trình bày trên màn hình. Color có giá trị từ 0 đến 15 ứng với 16 màu. Các hằng xác định màu nền và chữ cho biến Color như sau:

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

Ö **Ghi chú:** Ta có thể dùng các hằng giá trị trên bằng chữ hoặc số đều được. Ví dụ: *TextColor(4)* hoặc *TextColor(Red)* đều có ý nghĩa là chọn chữ màu đỏ. Chọn chữ màu xanh và chữ nhấp nháy: *TextColor(Green + Blink).*

12. Hàm KeyPressed: Boolean

Hàm kiểm tra xem có phím nào được ấn trên bàn phím hay không. Nếu có hàm trả về giá trị *True,* nếu không hàm cho giá trị *False.*

13. Hàm ReadKey: Char

Hàm này chờ đọc một ký tự từ bàn phím *(ký tự được nhập không được hiển thị trên màn hình).* Các phím trên bàn phím như *A, B, C,... 1, 2, 3, 4,..v.v.* chỉ tạo một mã khi được ấn, còn các phím chức năng như *F1, F2,..., Home, End, Alt, Ctrl, Ctrl - Home,...* tạo hai mã khi được ấn, trong đó mã thứ nhất có giá trị *0.* Để nhận biết một hay một tổ hợp phím bất kỳ được ấn, ta phải dùng một biến kiểu *Char* với hai lần thực hiện hàm *ReadKey* như sau:

```
Ch := ReadKey;
If Ch = #0 then Ch := Readkey;
```

Sau đây là một số phím đặc biệt và tổ hợp phím hay dùng:

Esc	*27*	**â**	*0/80*
Tab	*9*	**ß**	*0/75*
Enter	*13*	**à**	*0/77*
Home	*0/71*	*F1*	*0/59*
End	*0/79*	*F2*	*0/60*

PageUp	*0/73*	*F10*	*0/68*
PageDown	*0/81*	*Ctrl - F1*	*0/94*
á	*0/72*	*Ctrl - F2*	*0/95*

4 Ví dụ 1: Dịch chuyển con trỏ và in một số dòng chữ trên màn hình.

```
Uses  CRT;
Var  x, y : Integer;
Begin
   ClrScr;
   x := 20;
   y := 3;
   GotoXY(x + 2, y);
   Write( ' PASCAL ' ); { In tu cot 22 dong 3 }
   GotoXY(x - 2, y + 2);
   Write( ' BAN HAY DEN VOI ' ); { In tu cot 18 dong 5 }
   GotoXY(x, y + 3);
   Write( ' TURBO PASCAL ' ); { In tu cot 20 dong 6 }
   GotoXY(WhereX + 2, WhereY);
   Write( ' 7.0 ' ); { sau TURBO PASCAL in số 7.0 }
   Readln;
End.
```

4 Ví dụ 2: Nhận biết phím nào được ấn.

```
Uses  CRT;
Var  Ch : Char;
Begin
   Write( ' Ban hay an mot phim bat ky : ' );a
   Ch := ReadKey;
   If  Ch : = #0 then
      Begin
         Ch := Readkey;
         Writeln( ' Ban vua an mot phim dac biet co ma = ', Ord(Ch));
      End
   Else
      Writeln( ' Ban vua an mot phim co ma ASCII = ', Ord(Ch));
   Readln;
```

```
End.
```

4 Ví dụ 3: Viết chương trình hiển thị 16 dòng với nội dung bất kỳ, tại đầu mỗi dòng hiển thị số thứ tự của dòng đó đồng thời hiển thị màu của dòng đó theo số thứ tự *(theo bảng màu).*

```
Uses  CRT;
Var  i : Integer;
Begin
   For  i := 0  to  15  do
      Begin
         TextColor( i );
         Writeln( i, ' la ma so mau cua dong nay. ' );
      End;
   Readln;
End.
```

_____ o ² o _____

BÀI 8. KIỂU XÂU KÝ TỰ

I. Khai báo và các phép toán:

Xâu *(String)* là kiểu dữ liệu có cấu trúc dùng để xử lý các xâu ký tự. Kiểu *String* có nhiều điểm tương tự như kiểu mảng *(Array)* nhưng cũng có điểm khác nhau là: số ký tự trong một biến kiểu xâu có thể thay đổi còn số phần tử của kiểu mảng luôn cố định.

1. Khai báo kiểu xâu:

VAR

Tên_Biến : String[n];

Trong đó: *n* là số ký tự tối đa có thể có của xâu. Chiều dài tối đa của một xâu là 255. Nếu trong phần khai báo không ghi *[n]* thì xâu có độ dài mặc định là 255.

4 Ví dụ:

```
Var
  HoTen : String[30]; { HoTen có thể chứa tối đa 30 ký tự }
  St : String; { St có thể chứa tối đa 255 ký tự }
```

Với *St* là một xâu, để chỉ ra các ký tự thứ *i* của *St* ta viết *St[i].* Các *St[i]* đều có kiểu *Char*. Ví dụ: *St := 'ABCD';* thì lệnh *Write(St[3])* sẽ in ra ký tự *'C'.*

Cấu trúc của *String* như sau: Trong bộ nhớ nó chiếm số *Byte* bằng số ký tự tối đa, cộng với một byte đầu tiên *(tại vị trí s[0])* chứa ký tự mà mã thập phân *ASCII* của ký tự này sẽ cho biết xâu đó có độ dài bao nhiêu.

Chẳng hạn biến *HoTen* bên trên được gán giá trị:

HoTen := 'Ly Dong Giang';

Khi đó, độ dài xâu chỉ là 13, mặc dù độ dài cực đại cho phép là 30 như đã khai báo. Sau đây cấu trúc xâu *HoTen*:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17		26	27	28	29	30
Chr(13)	L	y		D	o	n	g		G	i	a	n	g	*	*	*	*	*.....	*	*	*	*	*

Ö Ghi chú: Ký tự * biểu diễn ký tự không xác định.

2. Nhập và in xâu ký tự:

Muốn in một xâu ký tự ta dùng lệnh *Write(St)* hoặc *Writeln(St).*

Lệnh *Readln(St)* sẽ đọc các ký tự cho xâu *St* với độ dài thực sự là số ký tự gõ vào từ bàn phím. Nếu ta gõ *< Enter >* luôn mà không nhập cho nó ký tự nào thì *St* là xâu rỗng.

4 Ví dụ:

```
Var  YourName, st1, st2 : String[40];
Begin
   Write( ' Please enter your name: ' );
   Readln(YourName);
   Writeln( ' Hello ', YourName + ' ! ' );
   st1 := ' Turbo Pascal ';
   st2 := ' Borland"s product is  ' + st1;
   Writeln(st2);
   Readln;
End.
```

3. Các phép toán trên xâu ký tự:

a. Phép gán:

Biến := Biểu_thức;

Đại lượng bên phải của lệnh phải được đặt giữa hai dấu nháy đơn nếu đó là xâu ở dạng hằng. Ta có thể sử dụng dấu cộng *(+)* để ghép các xâu khi gán. Ví dụ: *HoTen := 'Huynh Ngoc' + 'Nhan';*

b. Phép nối String:

Ký hiệu bằng dấu +.

4 Ví dụ: 'Turbo' + ' Pascal' = 'Turbo Pascal'

c. Các phép toán so sánh:

Khi so sánh hai xâu, các ký tự của hai xâu được so sánh từng cặp một từ trái qua phải theo giá trị trong bảng mã *ASCII.*

4 Ví dụ: Nếu so sánh:

'ABC' = 'ABC' có giá trị *True.*
'ABC' = 'AB' có giá trị là *False.*
'ABCD' < 'ABED' có giá trị là *True.*
'ABC' > 'AD' có giá trị là *False.*

II. Các thủ tục và hàm xử lý xâu ký tự:

1. Các thủ tục:

a. Delete(St , Pos, Num):

Trong đó:
- St *(String)*: Biến kiểu String.
- Pos *(Position)*: Biến kiểu nguyên.
- Num *(Number)*: Biến kiểu nguyên.

Công dụng: Thủ tục này dùng để xóa khỏi xâu *St* một số *Num* ký tự bắt đầu từ vị trí thứ *Pos.*

4 Ví dụ: Nếu *St = 'ABCDEFG';* thì:

Delete(St, 2, 4); **ð** làm cho *St = 'AFG'.*

Delete(St, 2, 10); **ð** làm cho *St = 'A'.*

Delete(St, 9, 3); **ð** làm cho *St = 'ABCDEFG'.*

b. Insert(St2, St1, Pos):

Trong đó:
- *St2* và *St1*: Biến kiểu *String.*
- *Pos*: Biến kiểu nguyên.

Công dụng: Thủ tục này dùng để chèn xâu *St2* vào xâu *St1* ở vị trí *Pos*. Ví dụ: Nếu *St := 'ABCD'* thì sau lệnh *Insert('TFG', St, 3)* ta nhận được *St := 'ABTFGCD'.*

Trường hợp *Pos* vượt quá chiều dài của *St1* thì *St2* sẽ được nối đuôi vào *St1.* Ví dụ: *St = 'ABCD'*, vậy lệnh *Insert('TFG', ST, 9);* sẽ làm cho *St = 'ABCDTFG'.*

c. Str(Value, St):

Trong đó:
- *Value*: Là một biểu thức nguyên hay thực có ghi dạng in ra.
- *St*: Biến kiểu String.

Công dụng: Thủ tục này dùng để đổi giá trị số *Value* thành kiểu xâu rồi gán cho *St.*

4 Ví dụ:

```
i := 1234;
Str(i:5, St); { ta được St = ' 1234' có 5 ký tự }
x :=123.5678901;
Str(x:10:5, St); { ta được St = ' 123.56789' }
```

d. Val(St, Var, Code):

Trong đó:
- *St*: Biểu thức kiểu String.
- *Var*: Là biến kiểu nguyên hay thực.
- *Code*: Biến kiểu nguyên.

Công dụng: Thủ tục này đổi xâu chữ *St (biểu diễn ở dạng số nguyên hay thực)* thành số và gán cho biến *Var*. *Code* là biến nguyên dùng để phát hiện lỗi: nếu phép biến đổi đúng thì *Code* có giá trị *0*, nếu sai do *St* không biểu diễn đúng số nguyên hay thực thì *Code* sẽ có giá trị bằng vị trí của ký tự sai trong xâu *St*. Ví dụ:

Giả sử: *St := '234'*, *i* và *e* là hai biến nguyên.
Val(St, i, e); { cho ta *i = 234* và *e = 0* }
Nếu *St := '21x'* thì *Val(St, i, e)* { cho ta i không xác định và *e = 3*, tức là ký tự thứ ba gây ra lỗi }

4 Ví dụ về một ứng dụng có sử dụng thủ tục Val để đọc số nguyên từ bàn phím. Bình thường ta dùng thủ tục *Readln(i)* để đọc số nguyên *i*. Song nếu nếu trong lúc nhập số, ta chẳng may gõ nhầm chữ cái vào thì máy dừng lại, có thể gây lãng phí thời gian. Thủ tục dưới đây có thể báo lỗi nếu ta nhập một số có chữ trong số đó.

```
Procedure InputInteger(Var  i : Integer);
  Var
    St : String[6];
    e : Integer;
  Begin
    Repeat
      Readln(St); { Nhập vào xâu số nguyên }
      Val(St, i, e); { Biến đổi và phát hiện lỗi }
      If  e <> 0  then
        Writeln(#7, ' Loi nhap lieu ! ');
    Until  e = 0;
  End;
```

2. Các hàm:

a. Length(St): cho ta độ dài của biểu thức xâu ký tự *St*. Ví dụ: với *St = "ABCDEFG'* thì *Length(St)* sẽ trả về giá trị *7*.

b. Copy(St, Pos, Num):

Trong đó: - *St*: Biểu thức kiểu xâu ký tự.
- *Pos,Num*: Biểu thức kiểu nguyên.

Hàm này trả về cho ta một xâu mới từ xâu *St*, hàm bắt đầu chép từ vị trí *Pos* và chép *Num* ký tự. Ví dụ: *St = 'ABCDEF'* thì lệnh *Copy(St, 3, 2) = 'CD'* và *Copy(St, 4, 10)* cho ta *'DEF'*.

Ö Ghi chú:

- Nếu *Pos + Num > Length(St)* thì hàm sẽ trả về các ký tự trong xâu *St.*
- Nếu *Pos > Length(St)* thì hàm *Copy* sẽ trả về cho ta một xâu rỗng.

c. Concat(St_1, St_2,..., St_n): Hàm này dùng để ghép tất cả các xâu ký tự *St1, St2,..., St_n* thành một xâu theo thứ tự các đối số cung cấp cho hàm.

Ö Ghi chú:

- Số lượng đối của hàm *Concat* phải *>= 2.*
- Nếu tổng số chiều dài các xâu *> 255* thì máy sẽ báo lỗi.
- Có thể dùng phép cộng *(+)* để ghép xâu ký tự. Ví dụ: *St :=Concat(St1,St2 + 'N');*

d. Pos(St1, St2):

Trong đó: *St1, St2* là biểu thức xâu ký tự.

Hàm này trả về số nguyên biểu diễn vị trí đầu tiên của *St1* gặp trong xâu *St2.* Nếu không tìm thấy thì *Pos = 0.*

4 Ví dụ: nếu *St := 'ABCDEFGBCD'* thì *Pos('DE',St) = 4, Pos('BCD',St) = 2, Pos('XY',St) = 0.*

4 Ví dụ 1: Viết chương trình nhập vào từ bàn phím một xâu ký tự và in ra màn hình xâu ký tự ngược tương ứng. Ví dụ: nhập *'TRUNG TAM CONG NGHE AVNET'* máy in ra *'TENVA EHGN GNOC MAT GNURT'.*

```
Program  DaoChuoi;
Uses  CRT;
Var
   Cau : String[80];
   i : Byte;
Begin
   Wite(' Nhap vao mot cau : ');
   Readln(Cau);
   For  i := Length(Cau)  DownTo  1  do
      Write(Cau[i]);
   Readln;
End.
```

4 Ví dụ 2: Hiển thị chuỗi con trong chuỗi mẹ được nhập từ bàn phím, vị trí và số ký tự hiển thị cũng được nhập từ bàn phím.

```
Program  SubString;
Uses  CRT;
Var
   St : String;
   Pos, Len : Byte;
Begin
   Wite(' Nhap vao mot chuoi : ');
   Readln(St);
   Wite(' Muon hien thi xau tu vi tri nao : ');
   Readln(Pos);
   Wite(' Do dai xau ky tu con : ');
   Readln(Len);
   Write(' Xau ky tu con la : ',Copy(St, Pos, Len));
   Readln;
End.
```

4 Ví dụ 3: Viết các hàm chuyển đổi xâu ký tự thành chữ hoa và chữ thường.

```
Function  ToUpper(s : String) : String;
Var  i : Byte;
Begin
   For  i := Length(s)  do
      s[i] := Upcase(s[i]);
   ToUpper := s;
End;
(*****************************)
Function  ToLower(s : String) : String;
Var  i : Byte;
Begin
   For  i := Length(s)  do
      If  s[i] In  ['A'..'Z']  then
         s[i] := Chr(Ord(s[i]) + 32);
   ToLower := s;
End;
```

BÀI 9. DỮ LIỆU KIỂU BẢN GHI VÀ KIỂU TỆP

I. Kiểu bản ghi:

1. Khái niệm và định nghĩa:

Các kiểu cấu trúc dữ liệu như kiểu mảng, tập hợp đều được tạo ra bằng một tập hợp các phần tử có cùng kiểu.

Để tạo ra một kiểu cấu trúc dữ liệu mới với các phần tử dữ liệu có kiểu khác nhau, người ta định nghĩa ra bản ghi *(Record). RECORD* là một cấu trúc bao gồm nhiều thành phần. Các thành phần có thể thuộc các kiểu dữ liệu khác nhau và được gọi là các trường *(Field),* mỗi trường đều được đặt tên.

Để mô tả một kiểu T có cấu trúc Record với danh sách các trường có tên là *S1, S2, ..., Sn* và có các mô tả kiểu tương ứng là trường có tên là *T1, T2, ... Tn* ta dùng cách viết như sau:

```
Type
   T = Record
      S1 : T1;
      S2 : T2;
      ...
      Sn : Tn;
   End;
```

Ví dụ: Mô tả thời gian DATE có ba trường: Ngày, Tháng, Năm

```
Type
   Date = Record
      Ngay:   1..31;
      Thang:  1..12;
      Nam:    Word;
   End;
```

4 Ví dụ: Để mô tả Nhân sự của phòng tổ chức, ta dùng các trường: HoDem, Ten, NgaySinh, Luong,... ở đây ta lấy ví dụ có 5 trường:

```
Type
   NhanSu = Record
      HoDem: String[20];
      Ten: String[7];
      NgaySinh: Date;
```

```
        Luong: Real;
        CoGiaDinh: Boolean;
    End;
    Var
      NV, NV1: NhanSu;
      DS: Array[1..100] of NhanSu;
      {Danh sach tren la kieu mang mo ta nhan su cua mot co quan co duoi 100
    nhan vien}
```

Ö Ghi chú: Ta có thể viết trực tiếp mô tả trường NgaySinh nếu như chưa có kiểu Date như sau:

```
    Type
      NhanSu = Record
        HoDem: String[20];
        Ten: String[7];
        NgaySinh: Record
          Ngay:     1..31;
          Thang:    1..12;
          Nam:      Word;
        End;
        Luong: Real;
        CoGiaDinh: Boolean;
    End;
```

2. Sử dụng Record:

Muốn truy cập một biến kiểu Record, ta phải truy cập theo thành phần của chúng. Cú pháp để truy cập đến một thành phần nào đó là:

<Tên biến Record>.<Tên trường>

4 Ví dụ:

```
NV.HoLot := 'Huynh Dinh';
NV.Ten := 'Can';
NV.NgaySinh.Ngay : = 4;
NV. NgaySinh.Thang := 2;
NV. NgaySinh. Nam := 1982;
NV.Luong := 500000;
NV.CoGiaDinh := False;
```

4 Ví dụ 1: Nhập lý lịch nhân viên của một cơ quan.

```
Uses CRT;
Type
   Date = Record
      Ngay:   1..31;
      Thang:  1..12;
      Nam:    Word;
   End;
   NhanSu = Record
      HoDem: String[20];
      Ten: String[7];
      NgaySinh: Date;
      Luong: Real;
      CoGiaDinh: Boolean;
End;
Var
   DS: Array[1..100] of NhanSu;
   i, SoNV: Byte;
   GD: Char;
Begin
   ClrScr;
   Writeln('          NHAP HO SO NHAN VIEN          ');
   Write(' So nhan vien tai co quan: ');
   Readln(SoNV);
   For  i:=1  to  SoNV  do
      Begin
         ClrScr;
         Write(' Ho dem: ');                Readln(DS[i].HoDem);
         Write(' Ho dem: ');                Readln(DS[i].Ten);
         Write(' Ngay sinh:  /  /');
         GotoXY(14,3);                      Readln(DS[i].NgaySinh.Ngay);
         GotoXY(17,3);                      Readln(DS[i].NgaySinh.Thang);
         GotoXY(20,3);                      Readln(DS[i].NgaySinh.Nam);
         Write(' Luong: ');                 Readln(DS[i].Luong);
         Write(' Co gia dinh (Y/N) ?: ');   Readln(GD);
```

```
            If  Upcase(GD) = 'Y'  then
               DS[i].CoGiaDinh := True
            Else
               DS[i].CoGiaDinh := False;
         End;
      Readln;
   End.
```

Ö Ghi chú:

- Các biến *Record* cùng kiểu có thể gán cho nhau. Ví dụ: *NV := NV1;* thay vì ta phải thực hiện:

NV.HoDem := NV1.HoDem;
NV.Ten := NV1.Ten;
.....

- Có thể dùng phép so sánh:

If NV = NV1 then Write(' Cung mot nhan vien ! ');

Hoặc:

If (NV.HoDem = NV1.HoDem) and (NV.Ten = NV1.Ten) then
Write(' Hai nhan vien cung ho ten !. ');

- Không được dùng các thao tác sau:

+ Các thủ tục đọc và ghi *(Read, Readln, Write, Writeln)* cho cả một biến kiểu Record như: *Readln(NV), Writeln(NV);*

+ Sử dụng các phép toán quan hệ như: *<, >, <=, >=.* Nhưng có thể sử dụng phép toán <> và = cho hai biến Record có cùng kiểu.

+ Tất cả các phép toán số học và logic.

3. Câu lệnh With:

Khi cần truy cập nhiều thành phần của một biến kiểu Record, ta có thể dùng câu lệnh *With* để chương trình được gọn hơn.

Cú pháp:

WITH <Biến kiểu Record> DO <Câu lệnh>

4 Ví dụ 1: Theo như ví dụ 1, ta có thể viết ngắn gọn hơn như sau:

```
Uses CRT;
Type
   Date = Record
      Ngay:   1..31;
```

```
      Thang:  1..12;
      Nam:    Word;
   End;
   NhanSu = Record
      HoDem: String[20];
      Ten: String[7];
      NgaySinh: Date;
      Luong: Real;
      CoGiaDinh: Boolean;
End;
Var
   DS: Array[1..100] of NhanSu;
   i, SoNV: Byte;
   GD: Char;
Begin
   ClrScr;
   Writeln('          NHAP HO SO NHAN VIEN          ');
   Write(' So nhan vien tai co quan: ');
   Readln(SoNV);
   For  i:=1  to  SoNV  do
      With  DS[i]  do
         Begin
            ClrScr;
               Write(' Ho dem: ');              Readln(HoDem);
               Write(' Ho dem: ');              Readln(Ten);
               Write(' Ngay sinh:  /  /');
               With  NgaySinh  do
                  Begin
                     GotoXY(14,3);          Readln(Ngay);
                     GotoXY(17,3);          Readln(Thang);
                     GotoXY(20,3);          Readln(Nam);
                  End;
               Write(' Luong: ');               Readln(Luong);
               Write(' Co gia dinh (Y/N) ?: ');  Readln(GD);
               If  Upcase(GD) = 'Y'  then
```

```
                CoGiaDinh := True
            Else
                CoGiaDinh := False;
        End;
    Readln;
End.
```

Ö Ghi chú: Như vậy chúng ta có thể lồng các chỉ thị With ... Do ... vào với nhau để truy nhập vào các trường ở sâu trong Record phức tạp như biến Ds[i]. Cú pháp như sau:

```
With A do
  With B do
    ......
```

Với A, B đều được mô tả là Record song B là một trường của A thì ta có thể có cách viết như sau:

```
With A do                          With A, B do
  With B do          ⟶               Begin
    Begin                              .....
      .....                          End;
    End;
```

4 Ví dụ 2: Đoạn chương trình ở ví dụ 1 có thể viết lại:

```
.....
For i:=1 to SoNV do
  With DS[i], NgaySinh do
    Begin
      ClrScr;
      Write(' Ho dem: ');              Readln(HoDem);
      Write(' Ho dem: ');              Readln(Ten);
      Write(' Ngay sinh:  /  /');
      GotoXY(14,3);                    Readln(Ngay);
      GotoXY(17,3);                    Readln(Thang);
      GotoXY(20,3);                    Readln(Nam);
      Write(' Luong: ');               Readln(Luong);
      Write(' Co gia dinh (Y/N) ?: '); Readln(GD);
      If Upcase(GD) = 'Y' then
```

```
            CoGiaDinh := True
        Else
            CoGiaDinh := False;
    End;
.....
```

4. Record có cấu trúc thay đổi:

Các kiểu Record trình bày trên là kiểu Record cố định vì số thành phần cũng như cấu trúc của Record là đã cố định. Bên cạnh đó Pascal còn *cho phép lập các Record có **một phần** cấu trúc thay đổi được.*

Trước hết, ta xét thí dụ sau: trong mục *NhanSu*, nếu ta xét thêm trường NgheNghiep thì sẽ có nhiều trường hợp xảy ra, chẳng hạn:

- *Công nhân : Cần ghi rõ ngành gì ? Bậc thợ mấy ?*
- *Kỹ sư : Ngành gì ? Trình độ thực tế ?*
- *Bác sĩ : Chuyên khoa gì ?*
- *Cá biệt : Không ghi gì thêm ?*

Tuy ta có thể lập một Record gồm đầy đủ các trường kể trên nhưng rất cồng kềnh *(trong khi đó có thể một người ở một thời điểm nào đó chỉ có một ngành nghề)* và chiếm nhiều ô nhớ.

Tiếp theo ta có thể lập ra bốn kiểu Record giống nhau phần đầu *(HoDem, Ten, NgaySinh, Luong, CoGiaDinh)* nhưng chỉ khác nhau phần cuối là nghề nghiệp *(NgheNghiep),* tức là sẽ có các trường tương ứng với bốn nghề khác nhau. Cách này cũng làm cồng kềnh chương trình vì ta phải dùng đến bốn kiểu Record.

Ngôn ngữ Pascal cho phép lập Record có dạng sau để tiết kiệm ô nhớ và cho phép linh hoạt sử dụng:

```
Type
   Nghe = (CongNhan, KySu, BacSi, CaBiet);
   Nganh = (KhaiThac, CoKhi, CheBien, Nuoi, KinhTe);
   Khoa = (Noi, Ngoai, Nhi, Phu);
   NhanSu = Record
      HoDem: String[20];
      Ten: String[7];
      NgaySinh: Date;
      Luong: Real;
```

```
        CoGiaDinh: Boolean;
        CASE  NgheNghiep: Nghe  Of
           CongNhan: (NganhCN: Nganh; BacTho: Byte);
           KySu: (NganhKS: Nganh; TrinhDoTT: (Kem, TB, kha, Gioi));
           BacSi: (ChuyenKhoa: Khoa);
           CaBiet: ();
        END; { Of  Record }
  Var  NV, NV1: NhanSu;
  Begin
     ...
     With  NV  do
        Begin
           HoDem := 'Vo Thanh';
           Ten := 'Chau';
           NgheNghiep := CongNhan;
           NganhCN := CoKhi;
           BacTho := 3;
        End;
     ...
     With  NV1  do
        Begin
           HoDem := 'Huynh Dinh';
           Ten := 'Can';
           NgheNghiep := KySu;
           NganhKS := KinhTe;
           TrinhDoTT := Kha;
        End;
     ...
  END.
```

F Giải thích minh hoạ trên:

- *HoDem, Ten, NgaySinh, CoGiaDinh* là các thành phần cố định của *Record NhanSu.*

- *NganhCN, NganhKS, BacTho, TrinhDoTT, ChuyenKhoa* là các thành phần thay đổi của *Record NhanSu.*

- Trong khai báo một kiểu Record, nếu có thành phần thay đổi thì *phải được đặt sau các thành phần cố định và chỉ được phép có một trường thay đổi*.
- Phần thay đổi nằm sau cùng trong danh sách và được bắt đầu bằng câu lệnh *CASE*. *(Phần thay đổi này lại có thể chứa Record khác có kiểu cấu trúc thay đổi).*

Ö Ghi chú:

- Phần thay đổi là một trường gọi là trường đánh dấu *(Tag Field)* và được dặt trong câu lệnh *CASE (Ví dụ trên là NgheNghiep).* Ứng với mỗi giá trị của trường đánh dấu, ta có các biến dạng của Record với danh sách các trường tương ứng được đặt sau các nhãn của lệnh *CASE* và toàn bộ danh sách này phải được đặt trong hai dấu ngoặc đơn *()* ngay cả khi nó rỗng như trường hợp *CaBiet* ở ví dụ trên.
- Trường mô tả phải là các kiểu đơn giản *(Byte, Integer, Word, LongInt, Real, Double, Char, Boolean)*.
- Tất cả các tên biến trong phần thay đổi đều bắt buột phải khác nhau. Theo ví dụ trên, *Nganh* trong hai trường hợp của *NgheNghiep* là *CongNhan* và *KySu* được ký hiệu bằng hai tên khác nhau là: *NganhCN* và *NganhKS.*

BÀI 10. DỮ LIỆU KIỂU TỆP

I. Khái niệm:

Khi giải các bài toán có nhiều và cần sử dụng nhiều lần về sau thì ta phải tổ chức dữ liệu lưu trữ trên đĩa *(dữ liệu kiểu tệp).* Khi kế thúc chương trình hoặc tắt máy thì dữ liệu kiểu tệp vẫn tồn tại trên đĩa.

Định nghĩa một kiểu tệp Tvới các phần tử có kiểu *KPT (Kiểu phần tử)* được viết trong phần mô tả kiểu với từ khoá *File Of* như sau:

```
TYPE
   T = FILE  OF  KPT;
```

4 Ví dụ:

```
Type
   FileReal = File  of  Real;

   Date = record
      Ngay: 1..31;
      Thang: 1..12;
      Nam: Word;
   End;

NhanSu = Record
   MaSo: Word;
   HoDem: String[20];
   Ten: String[7];
   NgaySinh: Date;
   Luong: Real;
End;

FnhanSu = File  Of  NhanSu;
Var
   F1: FileReal;
   F2: FNhanSu;
```

Ö Ghi chú:

- Kiểu phần tử của tệp có thể là bất kỳ kiểu dữ liệu nào ngoại trừ kiểu tệp.
- Biến tệp được khai báo bằng cách sử dụng một kiểu tệp đã được định nghĩa trước đó hoặc khai báo trực tiếp với mô tả kiểu. Ví dụ:

```
Var
```

F3: File Of Char;

F4: File Of Array[1..5] Of Integer;

- Biến tệp là một biến thuộc kiểu dữ liệu tệp. Một biến kiểu tệp đại diện cho một tệp. Việc truy cập dữ liệu ở một tệp được thể hiện qua các thao tác với thông số là biến tệp đại diện.

II. Cấu trúc và phân loại tệp:

Các phần tử của một *Array (Mảng)* hoặc *Record* có thể truy cập được tuỳ ý *(Random Access)* thông qua tên biến, chỉ số hoặc tên trường. Các phần tử của tệp không có tên và việc truy cập không thể tuỳ tiện được. Các phần tử của tệp được sắp xếp thành một dãy và ở mỗi thời điểm chương trình chỉ có thể truy nhập vào một phần tử của tệp thông qua giá trị của biến đệm *(Tampon Variable).* Biến đệm dùng để đánh dấu vị trí truy nhập hay còn gọi là cửa sổ của tệp. Ta có thể hình dung một tệp như là một cuộn phim chụp ảnh. Mỗi một ảnh là một phần tử và ống kính là cửa sổ để nhìn vào nên tại mỗi thời điểm chỉ nhìn thấy một ảnh. Sau mỗi lần chụp, cửa sổ sẽ nhìn vào ảnh ở vị trí kế tiếp.

Ta có thể dùng lệnh làm dịch chuyển cửa sổ sang vị trí tiếp theo hoặc về vị trí đầu tệp. Mỗi tệp đều được kết thúc bằng dấu hiệu đặc biệt để báo hiệu hết tệp, hay gọi là *EOF(F) (End Of File F).* Pascal có một hàm chuẩn *EOF* trả về giá trị kiểu Boolean với tham số là biến tệp để xem cửa sổ đã đặt vào vị trí kết thúc tệp đó chưa. Nếu chưa đến cuối tệp thì hàm *EOF* trả về giá trị *False.*

Việc phân loại tệp dựa trên việc bố trí các phần tử của tệp trong bộ nhớ ngoài và cách truy cập vào tệp: Tệp truy nhập tuần tự *(Sequential Access)* hoặc tệp truy nhập trực tiếp *(Direct Access).*

Đối với tệp truy nhập tuần tự việc đọc một phần tử bất kỳ của tệp phải đi qua các phần tử trước đó; muốn thêm một phần tử vào tệp, phải đặt cửa sổ vào vị trí cuối tệp. Bộ nhớ ngoài tương ứng với cấu trúc này là băng từ. Tệp truy nhập tuần tự đơn giản trong việc tạo lập hay xử lý nhưng kém tính linh hoạt.

Đối với tệp truy nhập trực tiếp, ta có thể đặt cửa sổ vào một vị trí bất kỳ của tệp. Bộ nhớ ngoài điển hình là đĩa từ *(do đầu từ khi đọc có thể được điều khiển đặt vào một chỗ bất kỳ trên đĩa tại mọi thời điểm).*

Tệp truy nhập trực tiếp chỉ được định nghĩa ở Turbo Pascal, Pascal chuẩn không có. Khi không nói rõ là tệp loại gì thì đó được mặc định là tệp truy nhập tuần tự.

III. Các thao tác trên tệp:

1. Mở tệp mới để cất dữ liệu:

Chương trình chỉ có thể lưu lại dữ liệu vào một tệp sau khi ta làm thủ tục mở tệp. Việc mở tệp được tiến hành với hai thủ tục đi liền nhau theo thứ tự:

Assign(FileVar, FileName)
ReWrite(FileVar);

Trong đó:

- FileVar:
- FileName: tên của tệp đặt trong thiết bị nhớ ngoài được đưa vào dạng một *String (quy tắc đặt tên tương tự hệ điều hành).* Ta nên đặt tên sao cho tên đó phản ánh được ý nghĩa hay bản chất, nội dung của tệp.

4 Ví dụ:

```
Assign(F1, 'HoSo.txt');    {Gán tên là HoSo.txt cho biến F1}
ReWrite(F1);               {Mở tệp HoSo.txt , tệp chưa có phần tử nào}
```

Sau khi mở tệp xong, tệp sẽ rỗng vì chưa có phần tử nào, cửa sổ của tệp sẽ không có giá trị xác định vì nó trỏ vào cuối tệp (EOF).

Ö Ghi chú: Khi mở tệp, nếu trên bộ nhớ ngoài (cùng đường dẫn) đã có sẵn tệp có tên trùng với tên tệp được mở thì nội dung cũ sẽ bị xóa.

2. Ghi các giá trị vào tệp với thủ tục Write:

Thủ tục Write sẽ đặt các giá trị mới vào tệp.

Cú pháp:

Write(FileVar, Item1, Item2, ..., ItemN);

Trong đó: *Item1, Item2, ..., ItemN:* là các giá trị cần ghi vào tệp.

4 Ví dụ: Ta cần ghi vào tệp *ChuCai.txt* các giá trị 'a'.. 'z', thực hiện như sau:

```
...
Assign(F1, 'ChuCai.txt');
ReWrite(F1);
For ch:= 'a' to 'z' do
Write(F1, ch);
...
```

4 Ví dụ 1: Tạo một tệp chứa các số nguyên từ 1 đến 100 với tên tệp trên đĩa là *"Nguyen.txt'.*

```
Program  TaoTepSoNguyen;
   Var   i: Integer;
         F: File  of  Integer;
   Begin
      Assign(F,'Nguyen.txt');
      ReWrite(F);
      For  i:= 1  to  100  do
         Write(F,i);
      Close(F);
   End.
```

Ö Ghi chú: Một tệp có thể được dùng làm tham số của chương trình con với lời khai báo bắt buộc phải sau chữ *Var* tức là tệp được dùng làm tham số biến.

3. Đọc dữ liệu từ một tệp đã có:

Đối với tệp tuần tự, ta không thể vừa ghi vừa đọc được cùng một lúc. Sau khi ghi dữ liệu vào tệp và đóng lại, ta có thể đọc lại các giá trị dữ liệu trong tệp.

Một chương trình muốn sử dụng các dữ liệu đã được chứa trong một têp, đầu tiên phải mở tệp đó ra để đọc, thủ tục sau nhằm mở một đọc:

Cú pháp:

Assign(FileVar, FileName);
Reset(FileVar);

Sau lệnh *Reset*, nếu tệp không rỗng thì cửa sổ tệp bao giờ cũng trỏ vào phần tử đầu tiên của tệp và chương trình sẽ sao chép phần tử của tệp được trỏ sang biến đệm cửa sổ. Nội dung tệp này không bị xóa. Nếu ta mở một tệp chưa tồn tại trên đĩa thì sẽ có lỗi.

Để đọc dữ liệu từ tệp, ta dùng thủ tục *READ* dạng sau:

Read(FileVar, Var1, Var2,..., VarN);

Trong đó: *Var1, Var2,..., VarN* là các biến có cùng kiểu thành phần của *FileVar*. Gặp lệnh này máy sẽ đọc các giá trị tại vị trí cửa sổ đang trỏ *(nếu có)* gán sang biến tương ứng cùng kiểu. Sau đó, cửa sổ dịch chuyển sang vị trí tiếp theo và đọc giá trị cho biến khác, cứ thế đọc cho đến biến *VarN. READ* chỉ có thể đọc giá trị của tệp để gán giá trị cho các biến.

Việc đọc một phần tử của tệp cần thỏa mãn điều kiện: phần tử đó không phải là phần tử cuối tệp tức là *EOF*. Do đó, trước khi muốn đọc tệp và gán cho biến X, cần phải thử xem tệp đó đã kết thúc chưa bằng câu lệnh:

If Not EOF(FileVar) Then Read(FileVar, X);

Hoặc nếu muốn đọc tất cả các phần tử của tệp:

```
While Not EOF(FileVar) Do
  Begin
    Read(FileVar, X);
    Xử lý biến x nếu cần;
    ...
  End;
```

Thực hiện xong ta phải đóng tệp với thủ tục sau:

Close(FileVar);

4 Ví dụ1: Giả sử đã tồn tại một tệp có tên là *Nguyen.txt* chứa các số kiểu Byte và có ít nhất ba phần tử. Thực hiện đọc ra giá trị thứ nhất và thứ ba của tệp và gán cho hai biến A, B tương ứng.

```
Program DocSo;
Var  A, B: Byte;
     F: File Of Byte;
  Begin
     Assign(F,'Nguyen.txt');
     Reset(F);
     Read(F,A);     {đọc một phần tử thứ nhất của tệp ra biến A}
     Read(F,B);     {đọc một phần tử thứ hai của tệp ra biến B}
     Read(F,B);     {đọc một phần tử thứ hai của tệp ra biến B}
                    {lúc này B không giữ giá trị thứ hai nữa}
     Close(F);
  End.
```

Vì đây là tệp có cấu trúc tuần tự nên muốn đọc phần tử thứ ba ta buộc phải đọc qua phần tử thứ hai.

Ba lần *Read(F,...)* ở trên có thể thay thế bằng một lệnh đọc duy nhất:

```
Read(F,A, B, B);
```

4 Ví dụ 2: Đọc tất cả các phần tử của một tệp chứa các số có Integer nào đó và ghi ra màn hình giá trị các số đó và cuối cùng ghi ra só phần tử của tệp.

```
Program  DocTepSo;
Uses  CRT;
Var   i, SoPT: Integer;
      F: File  Of  Byte;
      FileName: String;
   Begin
      ClrScr;
      Write('Tep can doc la gi ? (Tep so nguyen):');
      Readln(FileName);
      Assign(F, FileName);
      Reset(F);
      SoPT:= 0;
      While  Not  EOF(F)  Do
        Begin
          Read(F,i); {doc mot phan tu cua tep ra bien i}
          Write(i,'  ');
          Inc(SoPT); {dem so phan tu}
        End;
      Close(F);
      Writeln;
      Write('So phan tu cua tep ',FileName,' la ',SoPT);
      Readln
   End.
```

4. Tệp truy nhập trực tiếp:

Pascal chuẩn chỉ định nghĩa một kiểu tệp truy nhập tuần tự. Tuy nhiên các bộ nhớ ngoài như đĩa từ,... có thể cho phép tính toán tọa độ của một phần tử bất kỳ trong tệp (vì độ dài của các phần tử là như nhau), do đó có thể truy nhập trực tiếp vào một phần tử của tệp mặc dù cấu tạo logic của tệp vẫn là dạng tuần tự. Trong *Turbo Pascal*, để truy nhập trực tiếp vào phần tử của tệp, sử dụng thủ tục ***SEEK***.

Cú pháp:

Seek(FileVar, No);

Trong đó, *No* là số thứ tự của phần tử trong tệp *(phần tử đầu tiên của tệp được đánh số 0)*. Gặp thủ tục này, chương trình sẽ đặt cửa sổ của tệp vào phần tử thứ *No*. Tiếp theo muốn đọc phần tử đó ra thì dùng *Read*, nếu muốn đặt giá trị mới vào dùng *Write*.

4 Ví dụ: Giả sử tệp *Nguyen.txt* trên đĩa ở thư mục hiện hành đã chứa 100 số nguyên từ *1 đến 100*. Ta kiểm tra xem phần tử thứ hai *(đếm từ 0)* của tệp có giá trị bằng *3* không, nếu không thì sửa lại bằng một giá trị nhập từ bàn phím.

```
Var
   i: Byte;
   F: File  Of  Byte;
   Answer: Char;
Begin
   Assign(F,'Nguyen.txt');
   Reset(F);
   Seek(F,2); { Dat cua so tep vao vi tri thu 3}
   Read(F,i);
   Writeln('i = ',i);
   Write('Ban muon sua lai khong ?(C/K):');
   Readln(Answer);
   If  Answer  In['c','C']  Then
      Begin
         Seek(F,2);
         Write(' Ban muon sua lai bang bao nhieu ?');
         Readln(i);
         Write(F,i); { Thay doi gia tri cua phan tu hien tai }
      End;
   Close(F);
   Readln
End.
```

5. Các thủ tục và hàm xử lý tệp của Turbo Pascal:

a. Hàm FileSize(FileVar): Hàm cho giá trị biểu thị số phần tử của tệp *FileVar*. Hàm nhận giá trị 0 khi tệp rỗng.

b. Hàm FilePos(FileVar): cho biết vị trí hiện tại của con trỏ *(cửa sổ)* tệp FileVar.

Một tệp đã tồn tại chỉ có thể lớn thêm bằng cách ghi thêm các phần tử mới vào vị trí cuối cùng của tệp. Muốn đưa con trỏ đến vị trí cuối cùng của tệp ta thực hiện lệnh sau:

Seek(FileVar, FileSize(FileVar));

c. Thủ tục Erase(FileVar): Dùng để xóa tệp trên đĩa có tên ấn định với FileVar.

Ö **Chú ý:** Không được xóa tệp đang mở.

4 Ví dụ:

```
...
Write('Cho biet ten tep can xoa:');
Readln(FileName);
Assign(F, FileName);
Erase(F);
...
```

d. Thủ tục Rename(FileVar, Str): Dùng để thay đổi tên tệp với tên mới bằng biến *Str* kiểu *String*.

Ö **Ghi chú:** - Tên mới phải không trùng tên tệp nào có sẵn trên đĩa đang làm việc

- Không được đổi tên tệp đang mở.

4 Ví dụ: Muốn đổi tên tệp File1.dat thành File2.dat, thực hiện như sau:

```
Asign(F,'File1.dat');
Rename(F,'File1.dat');
```

6. Tệp văn bản (Text Files):

Trong Pascal có một kiểu tệp đã được định nghĩa trước, đó là tệp văn bản được định nghĩa với tên chuẩn Text.

Cú pháp khai báo:

F1, F2 :Text;

Thành phần cơ sở của tệp kiểu Text là ký tự. Tuy nhiên, văn bản có thể được cấu trúc thành các dòng, *mỗi dòng được kết thúc bởi dấu hiệu EOLN (End Of Line)*. Như vậy, muốn đọc và in ra từng dòng của tệp văn bản thì sử dụng dạng Text.

Tệp văn bản được kế tthúc bởi dấu End Of File, cụ thể với Turbo Pascal là *Ctrl-Z (^Z)* có mã ASCII = 26.

a. Hàm EOF(Var F: Text): Boolean. Hàm trả về giá trị False khi cửa sổ tệp chưa đến cuối tệp, ngược lại, cho giá trị True. Hàm này thường sử dụng để kiểm tra xem đã đọc hết tệp văn bản chưa. Ví dụ:

While not EOF(F) Do...

b. Hàm EOLN(Var F: Text): Boolean. Hàm trả về giá trị False khi cửa sổ tệp chưa đến điểm cuối dòng hoặc cuối tệp, ngược lại, cho giá trị True. Hàm này thường sử dụng để kiểm tra xem đã đọc đến cuối dòng chưa. Ví dụ:

While not EOLN(F) Do..

c. Ghi vào một tệp văn bản: Ta có thể ghi các giá trị kiểu *Integer, Real, Boolean, String* vào tệp văn bản bằng lệnh Write hoặc Writeln. Có ba dạng viết:

Write(FileVar, Item1, Item2,...,ItemN); (1)
Writeln(FileVar, Item1, Item2,...,ItemN); (2)
Write(FileVar); (3)

F Lệnh (1): Viết các giá trị *Item1, Item2,...,ItemN* là các hằng, biểu thức hay biến có kiểu đơn giản như: *Nguyên, Thực, Ký tự, Chuỗi, Logic* vào biến tệp FileVar.

F Lệnh (2): Tương tự như (1) nhưng có thêm dấu hiệu hết dòng vào tệp sau khi đã viết hết các giá trị *Item1, Item2,...,ItemN.*

F Lệnh (3): chỉ thực hiện việc đưa thêm dấu hiệu hết dòng vào tệp.

Ö Ghi chú: Từ câu lệnh (2) ta có thể chuyển sang viết như sau:

```
Begin
   Write(FileVar, Item1);
   ...
   Write(FileVar, Item2);
   Writeln(FileVar);
End;
```

4 Ví dụ: Thực hiện ghi vào một tệp các thông tin sau:

Chao cac ban den voi ngon ngu lap trinh Pascal
Trung tam Cong nghe Avnet

```
Var F: Text;
Begin
```

```
    Assign(F,'VanBan.txt');
    Rewrite(F);
    Writeln(F,'Chao cac ban den voi ngon ngu lap trinh Pascal');
    Writeln(F,'          Trung tam Cong nghe Avnet          ');
    Writeln(F,'          ------------------------           ');
    Writeln(F);
    Close(F);
End.
```

Ö Ghi chú: Trong lệnh Writeln,Write ta có thể hiển thị có quy cách như đã trình bày trước đây.

d. Đọc dữ liệu từ tệp văn bản:

Ta có thể đọc không những các ký tự từ tệp văn bản mà còn có thể đọc lại các số nguyên, thực, logic từ tệp văn bản thông qua các thủ tục:

Read(FileVar, Var1, Var2,..., VarN); *(1)*
Readln(FileVar, Var1, Var2,..., VarN); *(2)*
Readln(FileVar); *(3)*

Trong đó, *Var1, Var2,..., VarN* là các biến thuộc kiểu ký tự, nguyên, thực, logic, chuỗi. *Lệnh (1)* sẽ đọc nội dung một hay nhiều phần tử mà không chuyển cửa sổ tệp xuống dòng. *Lệnh (2)* đọc như lệnh (1) nhưng sẽ di chuyển cửa sổ tệp sang đầu dòng tiếp theo sau khi đã lần lượt đọc các biến tương ứng. *Lệnh (3)* đưa cửa sổ tệp sang đầu dòng tiếp theo mà không đọc gì cả.

4 Ví dụ: Chương trình sau đọc và in ra nội dung tệp văn bản *VanBan.txt* đã được tạo ra từ chương trình trên.

```
Program  Doc_File_Text;
  Uses  CRT;
  Var
    F: Text;
    Line:String[80];
  Begin
    ClrScr;
    Assign(F,'VanBan.txt');
    Reset(F);
    While  Not  EOF(F)  Do
      Begin
```

```
            Readln(F, Line);
            Writeln(Line);
        End;
    Close(F);
    Readln;
End.
```

e. Thủ tục thêm dòng:

Cú pháp:

Append(Var F: Text);

Lệnh Append mở tệp văn bản để ghi bổ sung các dòng, định vị cửa sổ tệp vào cuối tệp. Lần sử dụng kế tiếp với thủ tục Write hay Writeln sẽ thêm văn bản vào cuối tệp.

4 Ví dụ: Chương trình sau đây thêm hai dòng vào cuối tệp *VanBan.txt*.

```
Var F: Text;
Begin
    Assign(F,'Vanban.txt');
    Append(F);
    Writeln(F,'Day la dong thu nhat them vao.');
    Writeln(F,'Day la dong thu hai them vao.');
    Close(F);
End.
```

_____ o ² o _____

PHẦN BÀI TẬP THỰC HÀNH

: 1. Luyện tập căn bản: Khởi động chương trình Pascal và thực hiện:

1.1. Viết chương trình hiển thị lên màn hình nội dung sau :

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*                  Trung tam Cong nghe AVnet                  *
*                     74 - Tran Quoc Toan                     *
* * * * * * * * * * * * * * * * * * * * * * * * *
```

1.2. Viết chương trình hiển thị lên màn hình tam giác sau :

```
          *
        * * *
      * * * * *
    * * * * * * *
```

1.3. Viết chương trình hiển thị lên màn hình các biểu thức sau :

a. 5000 + 100 + 200
b. 645 + 350 - 345
c. 45 + 45 - 32

1.4. Viết chương trình để tính kết quả các biểu thức sau :

a. 5000 + 100 + 200
b. 645 + 350 - 345
c. 45 + 45 - 32

1.5. Chạy thử chương trình sau để tự rút ra nhận xét :

```
Program  BieuThuc;
Begin
   Write ( ' 45 + 756 + 16 = ' );
   Writeln (45 + 756 + 16 );
   Write ( ' 36 - 56 + 3 = ' );
   Writeln ( 36 - 56 + 3 );
   Readln;
End.
```

: 2. Bài tập đơn giản làm quen với các kiểu dữ liệu và một số hàm chuẩn của Pascal

2.1. Tìm chỗ sai trong chương trình sau:

```
Var  i, n : Integer;
b : Byte;
Begin
   n : =3;
   b := 278;
   i := b +n;
   Writeln(i);
End.
```

2.2. Viết chương trình nhập giá trị cho các biến từ bàn phím với kiểu của các biến là các kiểu dữ liệu đã được học, sau đó hiển thị mỗi giá trị của mỗi biến trên một dòng.

2.3. Viết chương trình đọc ký tự từ bàn phím, sau đó cho biết mã số của ký tự vừa nhập trong bảng mã *ASCII.*

2.4. Viết chương trình tính *Sin(x), Cos(x).* Trong đó, góc x được nhập từ bàn phím và được đo theo đơn vị Radian. (ta có thể chuyển đổi bằng cách: *Cos(x * Pi / 180)*)

2.5. Viết chương trình có sử dụng các hàm chuẩn của Turbo Pascal để tính giá trị:

- bình phương
- trị tuyệt đối
- căn bậc hai
- logric cơ số e *(e = 2.718)*
- hàm e mũ x *(e^x)*
- sau khi cắt bỏ phần thập phân
- làm tròn số

của x. Trong đó, x là một giá trị kiểu thực được nhập từ bàn phím.

2.6. Viết chương trình cho biết giá trị đứng trước, đứng sau của một giá trị x kiểu ký tự *(Char)* và y kiểu logic *(Boolean),* trong đó, *x* và *y* được nhập từ bàn phím.

: 3. Áp dụng các lệnh đơn giản

3.1. Chương trình sau cho kết quả gì?

```
Begin
   Writeln( False>True : 60 );
```

```
    Writeln( '1' > '2' );
    Readln;
End.
```

Thử lại trên máy để kiểm tra.

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

a) 5 + 3.0
b) 6/3 + 2 div 3
c) (10<=3) And (Not True And (12 div 3<=1))
d) (10*((45 mod 3) + 1)) / 6

Sau đó, viết chương trình thực hiện các phép tính trên *(hiển thị kết quả ở dạng có định quy cách).*

3.3. Viết chương trình tạo ra một thiệp mời dự sinh nhật. Trong đó, các giá trị lấy từ bàn phím gồm: *Họ tên người được mời, Ngày tổ chức tiệc, Địa điểm, Họ tên người mời.*

3.4. Viết chương trình tính tổng các chữ số của một số có 2 chữ số *(Hướng dẫn: sử dụng phép chia Div và Mod).*

3.5. Áp dụng phương pháp trên, viết chương trình tính tổng các chữ số của một số có *3, 4* chữ số.

3.6. Viết chương trình đổi một số nguyên được lấy từ bàn phím biểu diễn số giây thành giờ, phút, giây và hiển thị ở dạng *giờ : phút : giây*

3.7. Trong môi trường Turbo Pascal, để tạo ký tự chỉ cần ấn *Alt - 219 (các số 2, 1, 9 và gõ tại khu vực phím số).* Viết chương trình in lên màn hình từ **DA NANG** bằng ký tự

3.8. Viết chương trình tính giá trị biểu thức sau:

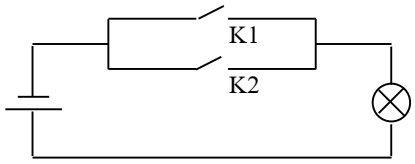
$$\frac{x^3 + Sin(b) - e^{0.0002345}}{5 + e^b + c(0.256 + x) - \sqrt{x + b}}$$

: 4. Bài tập cho các loại lệnh có cấu trúc

4.1. Bài tập cho cấu trúc lệnh If:

a. Viết chương trình để giải phương trình bậc hai $ax^2 + bx + c = 0$.

b. Viết chương trình mô tả sự hoạt động của mạch điện *(hình dưới)* khi *có hai công tắc mắc song song với nhau*, tức là cho biết *trạng thái sáng hay tối* của bóng đèn khi hai công tắc đóng hoặc ngắt. *(Hướng dẫn: Sử dụng các biến logic với phép toán OR).*



c. Nhập 3 số a, b, c tương ứng với 3 cạnh của một tam giác. Tính diện tích hình tam giác theo công thức:

$$s := \sqrt{p(p-a)(p-b)(p-c)}$$

d. Tính tiền thực lĩnh cho mỗi nhân viên trong xí nghiệp x theo công thức sau:

$$\text{Thực lĩnh} = \frac{(\text{Lương chính} * \text{Số ngày công})}{26} + (\text{Phụ cấp} - \text{Tạm ứng})$$

Với quy định: nghỉ quá 5 ngày sẽ bị trừ *20%* tổng thực lĩnh, làm thêm qá 3 ngày được tăng *10%* tổng thực lĩnh.

4.2. Bài tập cho cấu trúc lệnh Case:

a. Viết chương trình nhập một ký tự từ bàn phím, kiểm tra nó và:
- hiển thị *la so* nếu nó là số.
- hiển thị *la chu hoa* nếu nó là chữ hoa.
- hiển thị *la chu thuong* nếu nó là chữ thường.
- ngoài ra, hiển thị *Khong phai la so hoac chu cai.*

b. Viết chương trình đổi năm dương lịch *(dạng số)* thành năm âm lịch *(dạng chữ).* Ví dụ: nhập năm *2000* dương lịch máy cho biết năm âm lịch là *Canh Thìn*. ***(Hướng dẫn:*** sử dụng phép *MOD* giữa *năm* với 10 để lấy phần *Địa Can* và *MOD* giữa *năm* với 12 để lấy phần *Địa Chi*, số dư tương ứng sẽ được kết quả theo bảng sau:

Số dư (MOD 10)	Địa Can	Số dư (MOD 12)	Địa Chi
0	*Canh*	*0*	*Thân*
1	*Tân*	*1*	*Dậu*
2	*Nhâm*	*2*	*Tuất*
3	*Quý*	*3*	*Hợi*

4..... Giáp
5..... Ất
6..... Bính
7..... Đinh
8.....Mậu
9..... Kỷ

4..... Tý
5.....Sửu
6.....Dần
7..... Mẹo
8..... Thìn
9..... Tỵ
10.....Ngọ
11.....Mùi

c. Giải phương trình bậc hai $ax^2 + bx + c = 0$.

4.3. Bài tập cho cấu trúc vòng lặp For:

a. Viết chương trình nhập một số tự nhiên N từ bàn phím và tính:

$$e = 1 + 1/1! + 1/2! +... + 1/N!$$

b. Giải bài toán dân gian sau:

Vừa gà vừa chó.
Bó lại cho tròn.
Đếm đủ 100 chân.
Hỏi có mấy gà, mấy chó ?

c. Viết chương trình kiểm tra công thức sau đúng hay sai với mọi *N* dương được nhập từ bàn phím:

$$1 + 2 + 3 +... + N = N(N+1) / 2$$

d. Viết chương trình nhập vào chiều dài, chiều rộng của hình chữ nhật và in hình chữ nhật đó ra màn hình bằng các dấu *. Ví dụ: nhập *dài = 7, rộng = 3,* hình chữ nhật sẽ có dạng sau:

```
* * * * * * *
*         *
* * * * * * *
```

e. Viết chương trình tính **n!** trong đó, n là một số nguyên được nhập từ bàn phím (Hướng dẫn: ta nên khai báo biến để chứa kết quả là một biến kiểu **LongInt**).

4.4. Bài tập cho cấu trúc vòng lặp Repeat:

a. Viết chương trình làm các công việc sau: Tính diện tích hình chữ nhật, diện tích hình tam giác, hình tròn. Dùng lệnh **Repeat... Until** để lập một menu lựa chọn công việc theo mẫu:

TINH DIEN TICH CAC HINH

-1. Hinh chu nhat.
- 2. Hinh tam giac.
- 3. Hinh tron..
- 4. Ket thuc.

Lựa chọn một mục của menu bằng cách ấn số tương ứng, ấn phím số 4 máy dừng chương trình.

b. Viết chương trình nhập vào từ bàn phím lần lượt các số nguyên, dấu hiệu chấm dứt là số 0. Tính tổng và trung bình cộng của các số đã nhập.

c. Viết chương trình in ra bảng tính căn bậc hai của một trăm số nguyên dương đầu tiên.

4.5. Bài tập cho cấu trúc vòng lặp While:

a. Viết chương trình nhập vào từ bàn phím lần lượt các số nguyên, dấu hiệu chấm dứt là số 0. Tính tổng và trung bình cộng của các số đã nhập.

b. Viết chương trình tìm và hiển thị các số nguyên tố nhỏ hơn một số *n* được nhập từ bàn phím *(Số nguyên tố là số chỉ chia chẵn cho 1 và chính nó).*

c. Viết chương trình giả làm trò chơi xổ số như sau: Người chơi nhập 5 lần, mỗi lần một số nguyên tùy ý, máy kiểm tra nếu trong các số người chơi nhập vào có 3 số trở lên trùng với các số máy lấy ngẫu nhiên thì người đó thắng và ngược lại là thua. Nếu thua thì máy báo *Ban da thua !* ngược lại máy báo *Ban da thang !*

d. Viết chương trình nhập vào một ký tự *ch* bất kỳ, nếu nó là chữ số thì báo *ch la chu so,* nếu nó là chữ cái thì báo *ch la chu cai,* ngoài ra, báo *ch khong phai la so hoac chu cai* và thoát khỏi chương trình.

: 5. Bài tập cho dữ liệu kiểu đoạn con, liệt kê và kiểu mảng (Bài 5 và 6)

5.1. Viết chương trình nhập vào một dãy *n* số *a[1], a[2],..., a[n]* và in ra màn hình các thông tin sau:

- Tổng các phần tử của dãy.
- Số lượng số dương và tổng của các số dương của dãy.
- Số lượng số âm và tổng của các số âm của dãy.
- Trung bình cộng của dãy.

5.2. Viết chương trình nhập vào một dãy *n* số *a[1], a[2],..., a[n]* và in ra màn hình các thông tin sau:

- Số hạng dương lớn nhất của dãy và chỉ số *(vị trí)* của nó.
- Số hạng dương nhỏ nhất của dãy và chỉ số *(vị trí)* của nó.
- Số hạng âm lớn nhất của dãy và chỉ số *(vị trí)* của nó.
- Số hạng âm nhỏ nhất của dãy và chỉ số *(vị trí)* của nó.

5.3. Viết chương trình nhập vào mảng **a** gồm 10 phần tử nguyên, sau đó, nhập một giá trị *x*. Tìm trong mảng *a* nếu có phần tử nào *có giá trị bằng với x* thì hiển thị lên màn hình *vị trí* của nó trong mảng *a.*

5.4. Viết chương trình nhập vào một ma trận vuông, xuất màn ra màn hình ma trận đó và cho biết tổng các phần tử trên đường chéo chính.

: 7. Bài tập tạo thủ tục và hàm (Bài 7):

7.1. Viết một thủ tục dùng để vẽ hình vuông bằng dấu *. Chiều dài của cạnh hình vuông được nhập từ bàn phím. Gọi thực hiện thủ tục bởi chương trình chính.

7.2. Lập ba *thủ tục* tính diện tích hình tam giác, hình chữ nhật và hình tròn.

7.3. Lập ba *hàm* tính diện tích hình tam giác, hình chữ nhật và hình tròn.

7.4. Lập một hàm để kiểm tra một số có phải là số nguyên tố hay không. Sau đó, cho chương trình chạy liên tục và hỏi người dùng: *Bạn có tiếp tục không ?* cho đến khi người dùng nhập ký tự *k* hoặc *K* thì dừng lại.

7.5. Viết hàm để tính giá trị a^n. Trong đó, *a* và *n* là hai giá trị kiểu thực. *(Hướng dẫn:* $a^n = e^{n * ln(a)}$ *).*

7.6. Viết một hàm để tính giá trị *n!.*

: 8. Bài tập cho phần xử lý chuỗi (Bài 8):

8.1. Viết chương trình nhập vào một chuỗi và đếm trong chuỗi đó có bao nhiêu ký tự *'a', 'b'* và *'c' (kể cả 'A', 'B', 'C').*

8.2. Viết chương trình đếm trong một chuỗi được nhập từ bàn phím có bao nhiêu từ, giả sử mỗi từ cách nhau bằng một ký tự trắng *(tạm chấp nhận giữa hai từ không được nhập quá 1 ký tự trắng).*

8.3. Viết chương trình nhập vào một chuỗi *s*, sau đó, nhập vào một từ bất kỳ và kiểm tra trong chuỗi s nếu có từ đó thì xoá đi *(tại vị trí đầu tiên)*, nếu không tìm thấy từ đó trong s thì báo *Khong co tu nay trong chuoi vua nhap !*

8.4. Tương tự câu trên *(7.3)* nhưng nếu tìm thấy trong chuỗi s có bao nhiêu từ đó thì xoá hết.

8.5. Viết chương trình nhập vào từ bàn phím Họ và tên Việt Nam, sau đó in phần tên ra màn hình. Ví dụ: nhập *Phan Van Anh Tuan* thì in ra *Tuan*.

: BÀI TẬP TỔNG QUÁT

1. Tìm tất cả các số có 3 chữ số a, b, c sao cho tổng các lập phương của các chữ số bằng chính số đó.

$$abc = 100a + 10b + c = a^3 + b^3 + c^3$$

2. Tìm và in ra các số nguyên tố nhỏ hơn một số cho trước *n*.

3. Viết chương trình đếm số lần xuất hiện của từng loại ký tự từ *'A'* đến *'Z'* chứa trong một chuỗi được nhập từ bàn phím.

4. Nhập mảng hai chiều *A* gồm *m hàng* và *n cột*.

- Tìm giá trị lớn nhất và nhỏ nhất trên mỗi hàng, mỗi cột cùng với vị trí *(dòng, cột)* của giá trị này.

- Tìm phần tử có giá trị lớn nhất và nhỏ nhất của mảng *A* cùng với vị trí *(dòng, cột)* của hai phần tử này.

- Trong mảng *A* có bao nhiêu phần tử bằng với phần tử lớn nhất của mảng.

5. Viết chương trình nhập vào từ bàn phím một ma trận vuông và in ra màn hình tổng các phần tử trên đường chéo chính.

6*. Viết một chương trình dùng để giải các bài toán bằng cách tổ chức mỗi thủ tục để giải một bài toán và tạo menu để gọi thực hiện các thủ tục đó theo yêu cầu sau:

1. Giải phương trình bậc hai *($ax^2 + bx + c = 0$).*
2. Tính *Sin(x).*
3. Tính *Cos(x).*
4. Tính x^3.

Ö **Ghi chú:** Ngoài ra, học viên tự tìm thêm bài tập để thực hành.

_____ o ² o _____

MỤC LỤC

BÀI I. Giới thiệu ngôn ngữ pascal và các ví dụ đơn giản 1
I. Xuất xứ ngôn ngữ Pascal 1
II. Khởi động 1
III. Các phím chức năng cần biết của ngôn ngữ Pascal 2
IV. Cấu trúc một chương trình Pascal 2
1. Cấu trúc cơ bản 2
2. Phương pháp khai báo và tổ chức cấu trúc một chương trình Pascal 2
V. Các ví dụ đơn giản làm quen với ngôn ngữ Pascal 5

BÀI 2. Các khái niệm cơ bản của ngôn ngữ pascal 7
I. Các từ khoá (Key word) trong ngôn ngữ Pascal 7
II. Các kiểu dữ liệu cơ bản 7
1. Các kiểu dữ liệu dạng số nguyên 7
a. Kiểu Byte 7
b. Kiểu Integer 7
c. Kiểu Shortint 7
d. Kiểu Word 7
e. Kiểu Longint 7
2. Các kiểu dữ liệu dạng số có phần biểu diễn thập phân 7
a. Kiểu Single 7
b. Kiểu Real 7
c. Kiểu Double 7
3. Kiểu Char (ký tự) 8
4. Kiểu Logic 8
5. Kiểu String (chuỗi ký tự) 8
III. Các hàm xử lý dữ liệu cơ bản của ngôn ngữ Pascal 8
IV. Sử dụng hàm Random(n) để lấy một giá trị nguyên ngẫu nhiên 9

BÀI 3. Hằng số, biến số, biểu thức và câu lệnh đơn giản trong ngôn ngữ pascal .. 10
I. Hằng số 10
1. Khái niệm 10
2. Cú pháp khai báo 10
II. Biến số 11
1. Khái niệm 11
2. Cú pháp khai báo cho các biến 11

III. Biểu thức 12
IV. Câu lệnh đơn giản.....12
1. Lệnh gán 13
2. Lệnh Xuất 1 4
3. Lệnh Nhập.....17
BÀI 4. Các lệnh có cấu trúc trong ngôn ngữ pascal.....1 8
I. Lệnh ghép 1 8
II. Lệnh lựa chọn 19
1. Lệnh IF.....19
2. Lệnh CASE 21
III. Các câu lệnh lặp.....23
1. Câu lệnh FOR.....23
a. Dạng tiến.....23
b. Dạng lùi 2 4
2. Câu lệnh Repeat 2 5
3. Câu lệnh While.....27
IV. Các lệnh Goto, Break, Exit và Halt.....2 8
1. Lệnh Goto 2 8
2. Lệnh Break.....29
3. Lệnh Exit.....30
4. Lệnh Halt 30
Bài 5. Dữ liệu kiểu vô hướng liệt kê và kiểu đoạn con 31
I. Kiểu liệt kê.....31
II. Kiểu đoạn con.....32
Bài 6. Kiểu tập hợp và kiểu mảng 33
I. Kiểu tập hợp:.....33
1. Định nghĩa.....33
2. Các phép toán trên tập hợp 33
a. Phép toán quan hệ 33
b. Phép toán IN 3 4
c. Phép toán hợp, giao, hiệu.....3 4
II. Kiểu mảng 3 5
1. Khái niệm.....3 5
2. Khai báo mảng một chiều.....3 5
3. Truy cập các phần tử của mảng.....3 6

4. Mảng nhiều chiều 37
Bài 7. Chương trình con: Hàm và Thủ tục 40
I. Hàm và thủ tục 40
II. Biến toàn cục, biến cục bộ và việc truyền dữ liệu 42
III. Các hàm và thủ tục thường dùng của Unit CRT 44
1. Thủ tục ClrScr 44
2. Thủ tục ClrEOL 44
3. Thủ tục DelLine 45
4. Thủ tục InsLine 45
5. Thủ tục GotoXY(x, y: Byte) 45
6. Hàm WhereX: Byte 45
7. Hàm WhereY: Byte 45
8. Thủ tục Sound(Hz : Word) 45
9. Thủ tục NoSound 45
10. Thủ tục TextBackGround(Color : Byte) 45
11. Thủ tục TextColor(Color : Byte) 45
12. Hàm KeyPressed: Boolean 45
13. Hàm ReadKey: Char 45
Bài 8. Kiểu xâu ký tự 48
I. Khai báo và các phép toán 48
1. Khai báo kiểu xâu 48
2. Nhập và in xâu ký tư 48
3. Các phép toán trên xâu ký tư 49
a. Phép gán 49
b. Phép nối String 49
c. Các phép toán so sánh 49
II. Các thủ tục và hàm xử lý xâu ký tự 49
1. Các thủ tục 49
a. Delete(St , Pos, Num) 49
b. Insert(St2, St1, Pos) 50
c. Str(Value, St) 50
d. Val(St, Var, Code) 50
2. Các hàm 51
a. Length(St) 51

b. Copy(St, Pos, Num) 51
d. Pos(St1, St2) 52

Bài 9. Dữ liệu kiểu bản ghi và kiểu tệp 54
I. Kiểu bản ghi 54
1. Khái niệm và định nghĩa 54
2. Sử dụng Record 55
3. Câu lệnh With 57
4. Record có cấu trúc thay đổi 59

Bài 10. Dữ liệu kiểu tệp 62
I. Khái niệm 62
II. Cấu trúc và phân loại tệp 63
III. Các thao tác trên tệp 63
1. Mở tệp mới để cất dữ liệu 63
2. Ghi các giá trị vào tệp với thủ tục Write 64
3. Đọc dữ liệu từ một tệp đã có 65
4. Tệp truy nhập trực tiếp 67
5. Các thủ tục và hàm xử lý tệp của Turbo Pascal 68
a. Hàm FileSize(FileVar) 68
b. Hàm FilePos(FileVar) 68
c. Thủ tục Erase(FileVar) 68
d. Thủ tục Rename(FileVar, Str) 68
6. Tệp văn bản (Text Files) 69
a. Hàm EOF(Var F: Text): Boolean 69
b. Hàm EOLN(Var F: Text): Boolean 69
c. Ghi vào một tệp văn bản 69
d. Đọc dữ liệu từ tệp văn bản 70
e. Thủ tục thêm dòng 71

PHẦN BÀI TẬP THỰC HÀNH 72
: 1. Luyện tập căn bản 72
: 2. Bài tập đơn giản làm quen với các kiểu dữ liệu và một số hàm chuẩn của Pascal 72
: 3. Áp dụng các lệnh đơn giản 73
: 4. Bài tập cho các loại lệnh có cấu trúc 74
4.1. Bài tập cho cấu trúc lệnh If 74

4.2. Bài tập cho cấu trúc lệnh Case 7 5
4.3. Bài tập cho cấu trúc vòng lặp For 7 5
4.4. Bài tập cho cấu trúc vòng lặp Repeat 7 6
4.5. Bài tập cho cấu trúc vòng lặp While 7 6
: 5. Bài tập cho dữ liệu kiểu đoạn con, liệt kê và kiểu mảng 77
: 7. Bài tập tạo thủ tục và hàm 77
: 8. Bài tập cho phần xử lý chuỗi 7 8

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

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 & dể 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 tring*)
    { 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.

Vi du

Khai báo trong PASCAL

- Program tenchuongtrinh; { có thể có hoặc không}
  Program Chuongtrinhhinhtron;
- 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                                   { Thân chương trình chính }
      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.

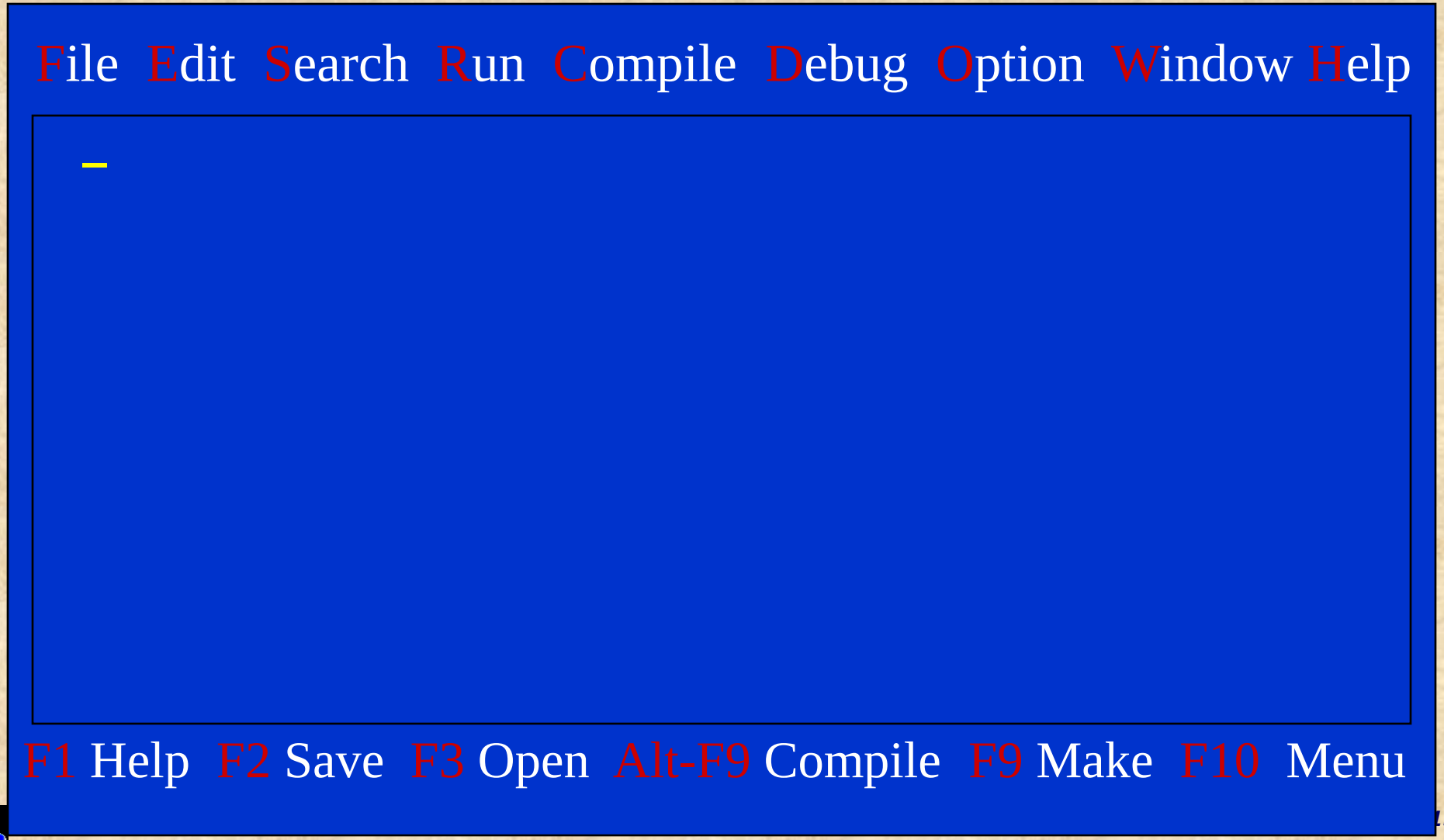
III. Môi trường làm việc của Turbo pascal (TP)

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



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 bộn 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 dự 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, mãng)
  - 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*

TỪ 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

Các phép toán số học đối với số nguyên

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**	$\pm 2,9.10^{-39} .. \pm 1,7.10^{38}$	**11-12**
Single	**4**	$\pm 1,5.10^{-45} .. \pm 3,4.10^{38}$	**7-8**
Double	**8**	$\pm 5,0.10^{-324} .. \pm 1,7.10^{308}$	**15-16**
Extended	**10**	$\pm 3,4.10^{-4932} .. \pm 1,1.10^{4932}$	**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 ASCCI. 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 ASCCI 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 thẻ là số hạng, thừa số, biểu thức đơn giản, hàm…

Ví dụ *CONST*

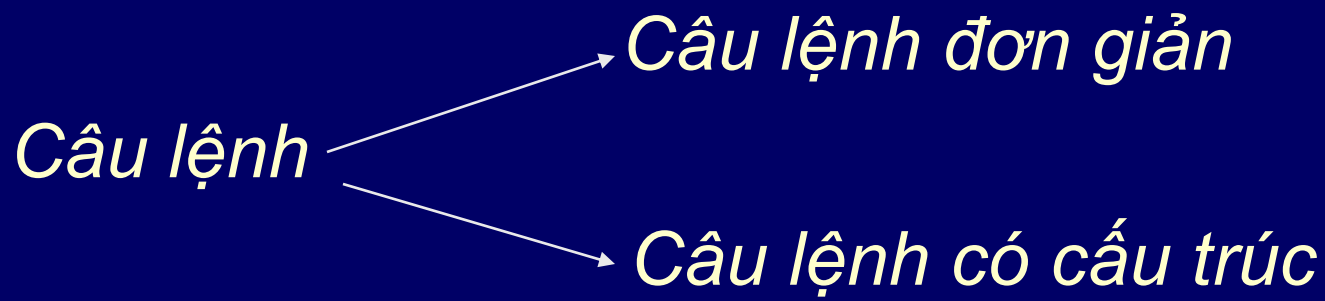
*3 + pi*sin(x);*

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)



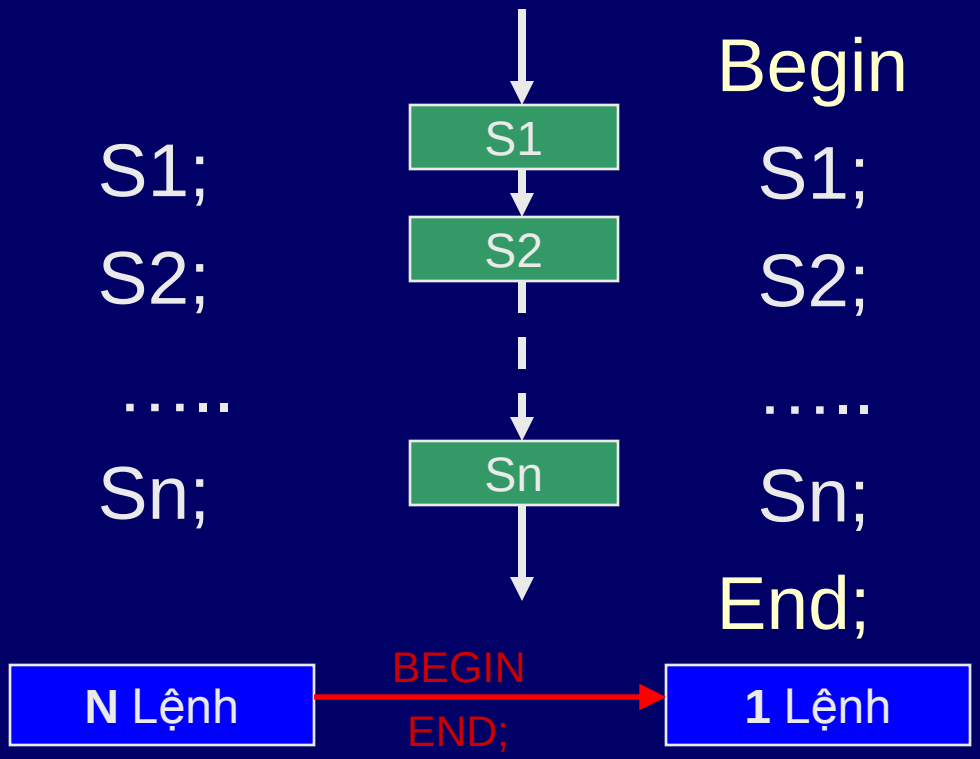
Câu lệnh đơn giản: Là các câu lệnh không chứa các lệnh khác read, write…

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

V. Khai báo hằng, 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;



Ví dụ

```
Begin
    t:=x;
    x:=y;
    y:=t;
    writeln('đã đổi xong');
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) *căn phải 16 kí tự*
- Viết ra kiểu số nguyên; với biến A=23123

  Write(A); Write(A:8); *căn phải 8 kí tự*
- Viết ra kiểu số thực; Với biến A=231.23

  Write(A); Write(A:8:3);

  Kết quả: 2.3123000000E+02 231.230
- Viết dữ liệu ra máy in:

  Write(Lst, Item1, Item2,….); Khi sử dụng lệnh này trong phần khai báo sử dụng lênh uses printer;

VI. Lệnh gán và thủ tục xuất nhập

2. Thủ tục vào dữ liệu:

- Read(biến1, biến2,…biếnN);
- ReadIn(biến1, biến2,…biếnN);
- ReadIn;

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; Lênh 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

1. Tổng quan

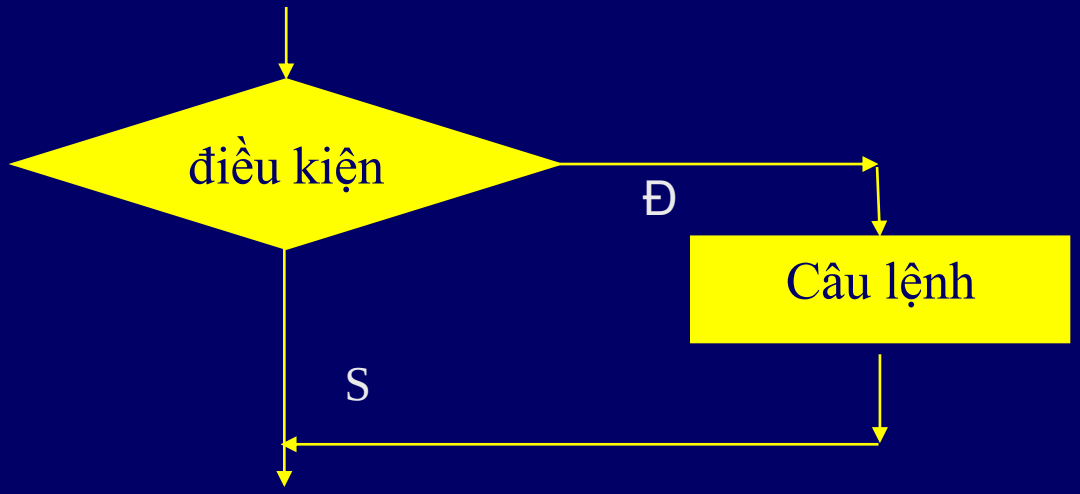
- **Lệnh điều khiển:** là những dòng lệnh dùng để điều khiển hoạt động của chương trình.
- **Các lệnh điều khiển cơ bản**

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;



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

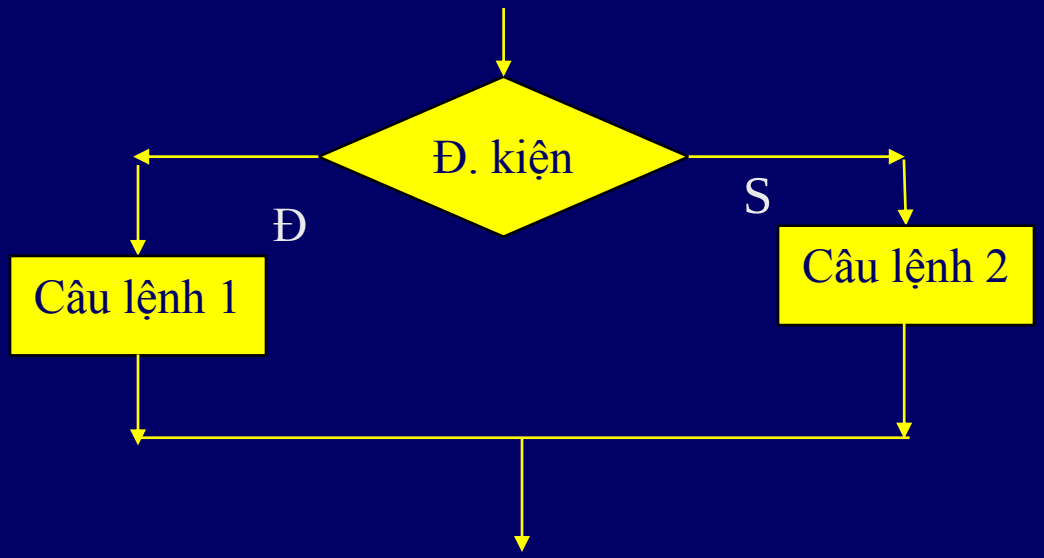
- Ví dụ

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

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

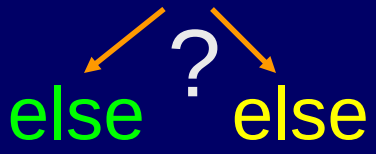
```
If Delta > 0 then
    begin
      X1:= (-b + sqrt(Delta))/2/a
      X2:= (-b - sqrt(Delta))/2/a
    end
 else Writeln('Còn xét tiếp');
```

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

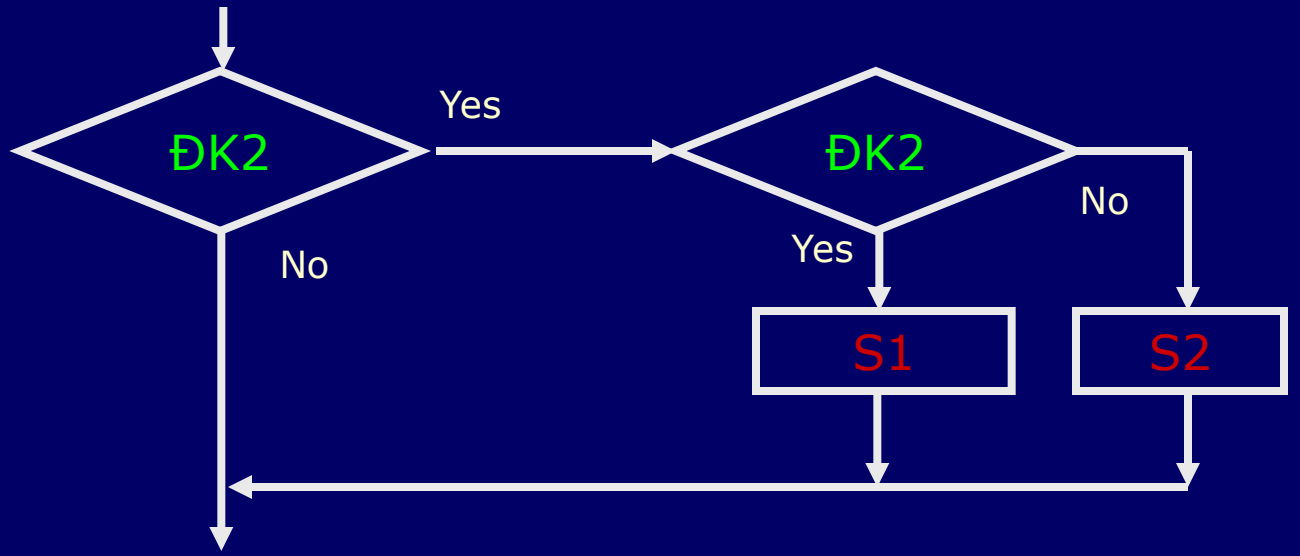
Học sinh viết chương trình giải phương trình bậc một, hai.

Trường hợp đặc biệt

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



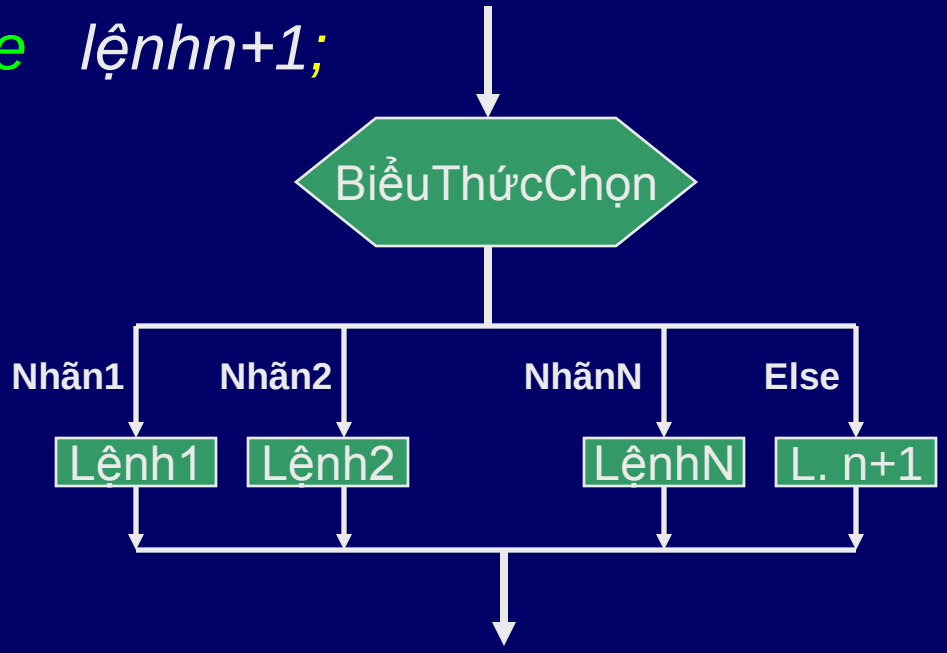
- ELSE sẽ thuộc về IF nào gần nhất chưa có ELSE



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:

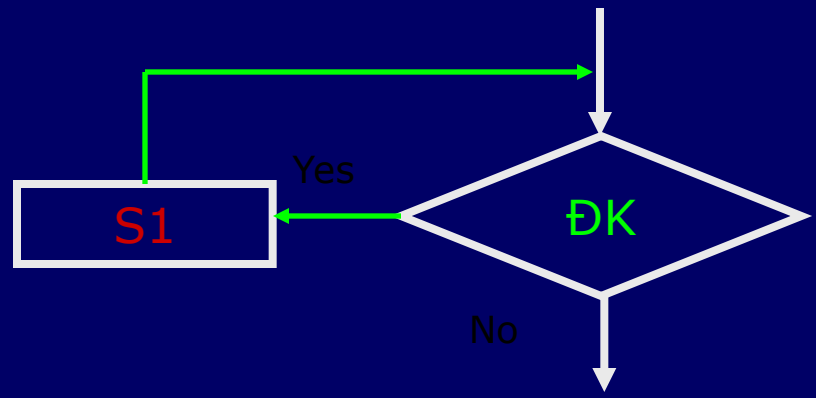
```
VAR
    songay, thang, nam: integer
BEGIN
    Write('Thang : '); Readln(thang);
    Write('nam : '); Readln(nam);
    case thang of
        4, 6, 9, 11: songay:=30;
        2: case nam mod 4 of
               0: songay:=29
               1, 2, 3: songay:=28
           End; {case nam}
    1, 3, 5, 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 lập lặp là không biết trước.
- Số lần lặp tối thiểu là 0 và tối đa là không xác định.
- *<u>Chú ý</u>: 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;
```

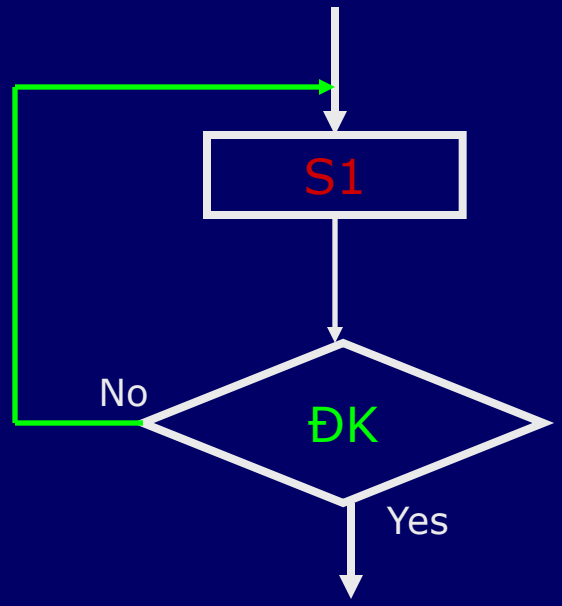


Phát biểu Repeat

- 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 lập lặp là không biết trước.
- Số lần lặp tối thiểu là 1 và tối đa là không xác định.
- *<u>Chú ý</u>: 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.
- White <ĐK> do statement;

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

- Repeat statements until <ĐK>;

=> Begin

statements;

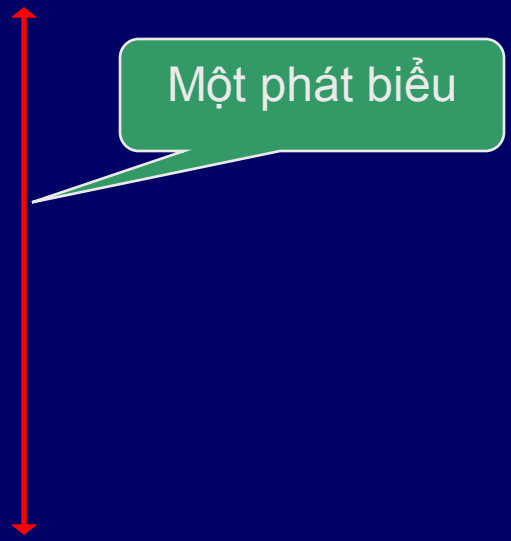
while NOT(<ĐK>) do begin

statements;

end;

End;

- Chú ý khả năng bị lặp vô tận.



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

- **Định nghĩa:** Kiểu miền con của một kiểu rời rạc là một miền trị của kiểu rời rạc đó (kiểu chủ) được xác định bằng 2 cận là 2 giá trị của kiểu chủ.
  Một giá trị thuộc kiểu miền
- **Mục tiêu sử dụng:**
  - Đối với một số compiler cho phép sinh mã tự động kiểm tra tính hợp lý của dự liệu thay vì phải tự kiểm tra bằng dòng lệnh. (chỉ thị là {$R+} vối Turbo Pascal).
  - Tiết kiệm bộ nhớ trong một số trường hợp.
  - Dùng trong khai báo các kiểu dữ liệu có cấu trúc khác như Array.
- Áp dụng được các phép toán của kiểu chủ cho kiểu miền con. Cần chú ý đến vùng giá trị kết quả để tránh runtime error
- Cú pháp : Tenkieu = canduoi . . Cantrên;

  diem = 1 . . 10;

  chu = ‘a’ .. ‘z’

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 tằng 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 dự 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

Vi 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 thống 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 dãy bất kỳ, tuy nhiên nếu dãy 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 nữa 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);
```

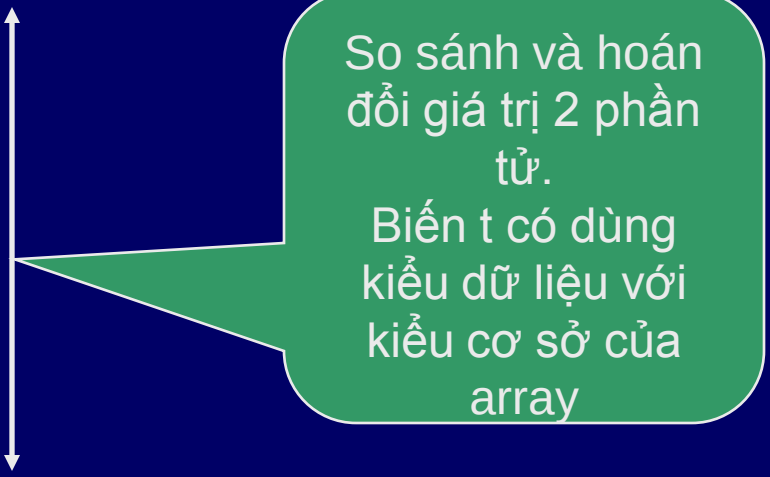
Bubble Sort

- 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 củ 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;
```



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 dield record tương ứng là một record đề có thể truy cập đến field bên trong dạng recordname.fieldrecord.field

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 muôn 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 <= : A<=B là TRUE nếu A bao hàm trong B
  - Phép toán >= : A>=B là TRUE nếu A chứa B
  - Phép toán IN : X IN A cho giá trị TRUE nếu trong A chứ 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>);

……..

CLOSE (<tên biến tệp>);

Ví dụ kiểu tệp

Tạo lập tệp SN.DAT bằng chương trình sau:

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

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 dư 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 phàn tử thức 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>);

Chú ý: Có thể dùng trình soạn thảo văn bản khác để tạo lập file văn bản

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 phàn tử thức 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 sec được mở trở 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 chươgnt rình 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.

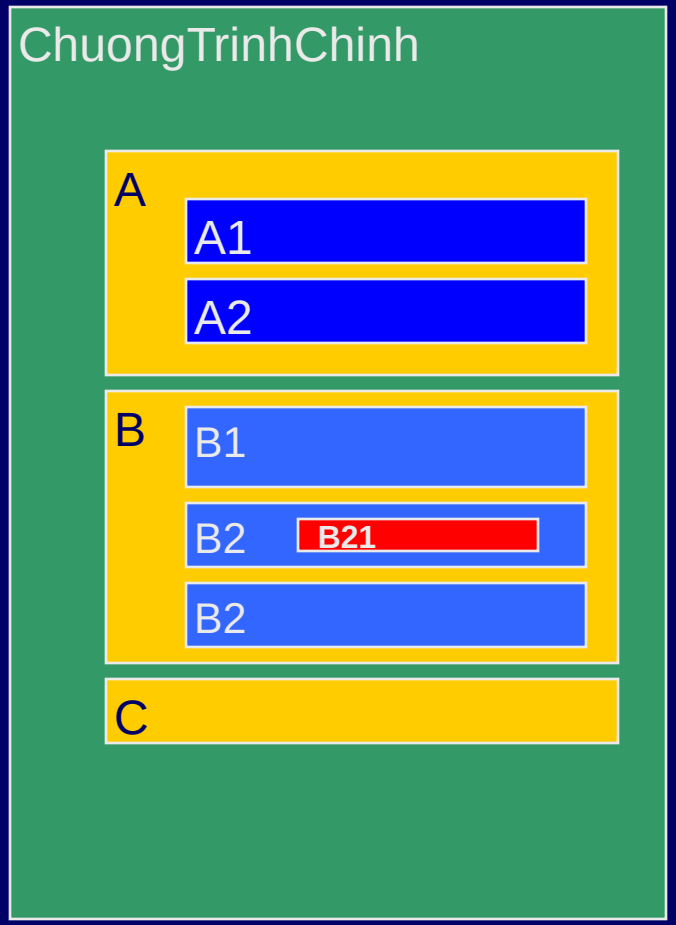
Ví dụ: procedure ABC (A: integer, var B: real, C:string);

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 trinh Pascal

- Định nghĩa Khối: Một khối (block) gồm 2 phần:
  - Phần khai báo với các khia 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.



Vấn đề tầm vực

- **Định nghĩa :** Tầm vực (Scope) của một đối tượng trong chương trình là vùng má nó được biết đến và có thể được sử dụng.
- **Tầm vực áp dụng trên các đối tương 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 tênX 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.

PHÒNG GIÁO DỤC ĐÀO TẠO DUY XUYÊN

GIÁO ÁN TRÌNH PASCAL

GV:LÊ VĂN CƯỜNG

TỔ: TOÁN - TIN

LỜI MỞ ĐẦU

Theo khung chương trình của Bộ Giáo Dục và Đào Tạo, **Ngôn ngữ Lập trình Pascal** là một phần quan trọng trong học phần Tin học Đại cương thuộc các khối ngành Khoa học Tự nhiên, đặc biệt là ngành Công nghệ Thông tin.

Nhằm đáp ứng yêu cầu học tập của học sinh, sinh viên bước đầu làm quen với công việc lập trình, chúng tôi đã biên soạn bộ **Giáo Trình Bài tập Pascal** nhằm giúp cho sinh viên có một tài liệu học tập, rèn luyện tốt khả năng lập trình, tạo nền tảng vững chắc cho các môn học tiếp theo trong chương trình đào tạo Cử nhân Công nghệ Thông tin .

Giáo trình bai gồm rất nhiều bài tập từ đơn giản đến phức tạp. Các bài tập này được biên soạn dựa trên khung chương trình giảng dạy môn **Tin học Đại cương**. Bên cạch đó, chúng tôi cũng bổ sung một số bài tập dựa trên cơ sở một số thuật toán chuẩn với các cấu trúc dữ liệu được mở rộng nhằm nâng cao kỹ năng, phương pháp lập trình cho sinh viên.

Nội dung của giáo trình được chia thành 10 chương. Trong mỗi chương đều có phần tóm tắt lý thuyết, phần bài tập mẫu và cuối cùng là phần bài tập tự giải để bạn đọc tự mình kiểm tra những kiến thức và kinh nghiệm đã học. Trong phần bài tập mẫu, đối với những bài tập khó hoặc có thuật toán phức tạp, chúng tôi thường nêu ra ý tưởng và giải thuật trước khi viết chương trình cài đặt.

Xin chân thành cảm ơn các đồng nghiệp ở Khoa Công nghệ Thông tin Trường Đại học Khoa học Huế đã giúp đỡ, đóng góp ý kiến để hoàn chỉnh nội dung giáo trình này.

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

Duy Xuyên, Tháng 04 Năm 2012

CÁC TÁC GIẢ

Chương 1

CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ LẬP TRÌNH PASCAL

Pascal là một ngôn ngữ lập trình bậc cao do Niklaus Wirth, giáo sư điện toán trường Đại học kỹ thuật Zurich (Thụy Sĩ) đề xuất năm 1970. Ông lấy tên Pascal để kỷ niệm nhà toán học và nhà triết học người Pháp nổi tiếng Blaise Pascal.

1. Các tập tin cần thiết khi lập trình với Turbo Pascal

Để lập trình được với Turbo Pascal, tối thiểu cần 2 file sau:

- **TURBO.EXE**: Dùng để soạn thảo và dịch chương trình.
- **TURBO.TPL**: Thư viện chứa các đơn vị chuẩn để chạy với TURBO.EXE.

Ngoài ra, muốn lập trình đồ hoạ thì phải cần thêm các tập tin:

- **GRAPH.TPU**: Thư viện đồ hoạ.
- ***.BGI**: Các file điều khiển các loại màn hình tương ứng khi dùng đồ hoạ.
- ***.CHR**: Các file chứa các font chữ đồ họa.

2. Các bước cơ bản khi lập một chương trình Pascal

Bước 1: Soạn thảo chương trình.

Bước 2: Dịch chương trình (nhấn phím **F9**), nếu có lỗi thì phải sửa lỗi.

Bước 3: Chạy chương trình (nhấn phím **Ctrl-F9**).

3. Cấu trúc chung của một chương trình Pascal

```
{ Phần tiêu đề }
PROGRAM  Tên_chương_trình;
{ Phần khai báo }
USES     ......;
CONST  .....;
TYPE     .......;
VAR      ........;
PROCEDURE  .............;
FUNCTION ..............;
.................
{ Phần thân chương trình }
BEGIN
   ...........
END.
```

Ví dụ 1: Chương trình Pascal đơn giản nhất

```
BEGIN
   Write('Hello World!');
END.
```

Ví dụ 2:

```
Program  Vidu2;
Const  PI=3.14;
Var  R,S:Real;
Begin
   R:=10;          {Bán kính đường tròn}
   S:=R*R*PI;    {Diện tích hình tròn}
   Writeln('Dien tich hinh tron = ', S:0:2);  { In ra màn hình }
```

```
    Readln;
  End.
```

4. Một số phím chức năng thường dùng

- **F2**: Lưu chương trình đang soạn thảo vào đĩa.
- **F3**: Mở file mới hoặc file đã tồn tại trên đĩa để soạn thảo.
- **Alt-F3**: Đóng file đang soạn thảo.
- **Alt-F5**: Xem kết quả chạy chương trình.
- **F8**: Chạy từng câu lệnh một trong chương trình.
- **Alt-X**: Thoát khỏi Turbo Pascal.
- **Alt-<Số thứ tự của file đang mở>**: Dịch chuyển qua lại giữa các file đang mở.
- **F10**: Vào hệ thống Menu của Pascal.

5. Các thao tác cơ bản khi soạn thảo chương trình

5.1. Các phím thông dụng

- **Insert**: Chuyển qua lại giữa chế độ đè và chế độ chèn.
- **Home**: Đưa con trỏ về đầu dòng.
- **End**: Đưa con trỏ về cuối dòng.
- **Page Up**: Đưa con trỏ lên một trang màn hình.
- **Page Down**: Đưa con trỏ xuống một trang màn hình.
- **Del**: Xoá ký tự ngay tại vị trí con trỏ.
- **Back Space (←)**: Xóa ký tự bên trái con trỏ.
- **Ctrl-PgUp**: Đưa con trỏ về đầu văn bản.
- **Ctrl-PgDn**: Đưa con trỏ về cuối văn bản.
- **Ctrl-Y**: Xóa dòng tại vị trí con trỏ.

5.2. Các thao tác trên khối văn bản

- Chọn khối văn bản: **Shift + <Các phím**
- **Ctrl-KY**: Xoá khối văn bản đang chọn
- **Ctrl-Insert**: Đưa khối văn bản đang chọn vào Clipboard
- **Shift-Insert**: Dán khối văn từ Clipboard xuống vị trí con trỏ.

6. Các thành phần cơ bản của ngôn ngữ Pascal

6.1. Từ khóa

Từ khoá là các từ mà Pascal dành riêng để phục vụ cho mục đích của nó. (Chẳng hạn như: **BEGIN, END, IF, WHILE,...**)

Chú ý: *Với Turbo Pascal 7.0 trở lên, các từ khoá trong chương trình sẽ được hiển thị khác màu với các từ khác*.

6.2. Tên (định danh)

Định danh là một dãy ký tự dùng để đặt tên cho các hằng, biến, kiểu, tên chương trình con... Khi đặt tên, ta phải chú ý một số điểm sau:

- Không được đặt trùng tên với từ khoá
- Ký tự đầu tiên của tên không được bắt đầu bởi các ký tự đặc biệt hoặc chữ số.
- Không được đặt tên với ký tự space,các phép toán.

Ví dụ: Các tên viết như sau là sai

1XYZ Sai vì bắt đầu bằng chữ số.
#LONG Sai vì bắt đầu bằng ký tự đặc biệt.
FOR Sai vì trùng với từ khoá.
KY TU Sai vì có khoảng trắng (space).
LAP-TRINH Sai vì dấu trừ (-) là phép toán.

6.3. Dấu chấm phẩy (;)

Dấu chấm phẩy được dùng để ngăn cách giữa các câu lệnh. Không nên hiểu dấu chấm phẩy là dấu kết thúc câu lệnh.

Ví dụ:

FOR i:=1 TO 10 DO Write(i);

Trong câu lệnh trên, lệnh Write(i) được thực hiện 10 lần. Nếu hiểu dấu chấm phẩy là kết thúc câu lệnh thì lệnh Write(i) chỉ thực hiện 1 lần.

6.4. Lời giải thích

Các lời bàn luận, lời chú thích có thể đưa vào bất kỳ chỗ nào trong chương trình để cho người đọc dễ hiểu mà không làm ảnh hưởng đến các phần khác trong chương trình. Lời giải thích được đặt giữa hai dấu ngoạc { và } hoặc giữa cụm dấu (* và *).

Ví dụ:

Var a,b,c:Rea; **{***Khai báo biến***}**
Delta := b*b – 4*a*c; **(*** *Tính delta để giải phương trình bậc 2* ***)**

BÀI TẬP THỰC HÀNH

1. Khởi động Turbo Pascal.
2. Nhập vào đoạn chương trình sau:

```
Uses Crt;
Begin
   Writeln('************************************************************');
   Writeln('*  CHUONG TRINH PASCAL DAU TIEN CUA TOI  *');
   Writeln('*                    Oi! Tuyet voi!...                     *');
   Writeln('************************************************************');
   Readln;
End.
```

3. Viết chương trình in ra màn hình các hình sau:

```
      *              ********              *******
     ***             **      **          **      **
    ** **            **      **          **
   **   **           ********            * *
  *********           **      **          **
 **       **         **      **          **      **
**         **        ********              ********
```

Chương **2**

CÁC KIỂU DỮ LIỆU CƠ BẢN
KHAI BÁO HẰNG, BIẾN, KIỂU, BIỂU THỨC VÀ CÂU LỆNH

I. CÁC KIỂU DỮ LIỆU CƠ BẢN

1. Kiểu logic

- Từ khóa: **BOOLEAN**
- miền giá trị: **(TRUE, FALSE)**.
- Các phép toán: phép so sánh (=, <, >) và các phép toán logic: AND, OR, XOR, NOT.

Trong Pascal, khi so sánh các giá trị boolean ta tuân theo qui tắc: FALSE < TRUE.

Giả sử A và B là hai giá trị kiểu Boolean. Kết quả của các phép toán được thể hiện qua bảng dưới đây:

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

2. Kiểu số nguyên

2.1. Các kiểu số nguyên

Tên kiểu	Phạm vi	Dung lượng
Shortint	-128 [redacted] 27	1 byte
Byte	0 [redacted] 55	1 byte
Integer	-32768 [redacted] 2767	2 byte
Word	0 [redacted] 5535	2 byte
LongInt	-2147483648 [redacted] 147483647	4 byte

2.2. Các phép toán trên kiểu số nguyên

2.2.1. Các phép toán số học:

+, -, *, / (phép chia cho ra kết quả là số thực).

Phép chia lấy phần nguyên: **DIV** (Ví dụ : 34 DIV 5 = 6).

Phép chia lấy số dư: **MOD** (Ví dụ: 34 MOD 5 = 4).

2.2.2. Các phép toán xử lý bit:

Trên các kiểu ShortInt, Integer, Byte, Word có các phép toán:

[redacted] NOT, AND, OR, XOR.

A	B	A AND B	A OR B	A XOR B	NOT A
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

[redacted] SHL (phép dịch trái): a SHL n [redacted] 2^n

[redacted] SHR (phép dịch phải): a SHR n [redacted] DIV 2^n

3. Kiểu số thực

3.1. Các kiểu số thực:

Tên kiểu	Phạm vi	Dung lượng
Single	1.5 [redacted] 0^{-45} [redacted] 4 [redacted] 0^{+38}	4 byte
Real	2.9 [redacted] 0^{-39} [redacted] 7 [redacted] 0^{+38}	6 byte

Double	5.0×10^{-324} ... $7 \times 10^{+308}$	8 byte
Extended	3.4×10^{-4932} ... $1 \times 10^{+4932}$	10 byte

Chú ý: Các kiểu số thực Single, Double và Extended yêu cầu phải sử dụng chung với bộ đồng xử lý số hoặc phải biên dich chương trình với chỉ thị **{$N+}** để liên kết bộ giả lập số.

3.2. Các phép toán trên kiểu số thực: **+, -, *, /**

Chú ý: *Trên kiểu số thực không tồn tại các phép toán DIV và MOD.*

3.3. Các hàm số học sử dụng cho kiểu số nguyên và số thực:

SQR(x): Trả về x^2
SQRT(x): Trả về căn bậc hai của x (x...)
ABS(x): Trả về |x|
SIN(x): Trả về sin(x) theo radian
COS(x): Trả về cos(x) theo radian
ARCTAN(x): Trả về arctang(x) theo radian
LN(x): Trả về ln(x)
EXP(x): Trả về e^x
TRUNC(x): Trả về số nguyên gần với x nhất nhưng bé hơn x.
INT(x): Trả về phần nguyên của x
FRAC(x): Trả về phần thập phân của x
ROUND(x): Làm tròn số nguyên x
PRED(n): Trả về giá trị đứng trước n
SUCC(n): Trả về giá trị đứng sau n
ODD(n): Cho giá trị TRUE nếu n là số lẻ.
INC(n): Tăng n thêm 1 đơn vị (n:=n+1).
DEC(n): Giảm n đi 1 đơn vị (n:=n-1).

4. Kiểu ký tự

- Từ khoá: **CHAR.**
- Kích thước: 1 byte.
- Để biểu diễn một ký tự, ta có thể sử dụng một trong số các cách sau đây:
  - Đặt ký tự trong cặp dấu nháy đơn. Ví dụ 'A', '0'.
  - Dùng hàm CHR(n) (trong đó n là mã ASCII của ký tự cần biểu diễn). Ví dụ CHR(65) biểu diễn ký tự 'A'.
  - Dùng ký hiệu #n (trong đó n là mã ASCII của ký tự cần biểu diễn). Ví dụ #65.
- Các phép toán: =, >, >=, <, <=,<>.

*** Các hàm trên kiểu ký tự:**

- **UPCASE(ch)**: Trả về ký tự in hoa tương ứng với ký tự ch. Ví dụ: UPCASE('a') = 'A'.
- **ORD(ch)**: Trả về số thứ tự trong bảng mã ASCII của ký tự ch. Ví dụ ORD('A')=65.
- **CHR(n)**: Trả về ký tự tương ứng trong bảng mã ASCII có số thứ tự là n. Ví dụ: CHR(65)='A'.
- **PRED(ch)**: cho ký tự đứng trước ký tự ch. Ví dụ: PRED('B')='A'.
- **SUCC(ch)**: cho ký tự đứng sau ký tự ch. Ví dụ: SUCC('A')='B'.

II. KHAI BÁO HẰNG

- Hằng là một đại lượng có giá trị không thay đổi trong suốt chương trình.
- Cú pháp:

CONST<Tên hằng> = <Giá trị>;

hoặc:

CONST<Tên hằng>: = <Biểu thức hằng>;

Ví dụ:

```
CONST Max = 100;
```

```
Name = 'Tran Van Hung';
Continue = FALSE;
Logic = ODD(5);     {Logic =TRUE}
```

Chú ý: *Chỉ các hàm chuẩn dưới đây mới được cho phép sử dụng trong một biểu thức hằng*:

ABS CHR HI LO LENGTH ODD ORD
PTR ROUND PRED SUCC SIZEOF SWAP TRUNC

III. KHAI BÁO BIẾN

- Biến là một đại lượng mà giá trị của nó có thể thay đổi trong quá trình thực hiện chương trình.
- Cú pháp:

VAR <Tên biến>[,<Tên biến 2>,...] : <Kiểu dữ liệu>;

Ví dụ:

```
VAR     x, y: Real;  {Khai báo hai biến x, y có kiểu là Real}
        a, b: Integer;  {Khai báo hai biến a, b có kiểu integer}
```

Chú ý: *Ta có thể vừa khai báo biến, vừa gán giá trị khởi đầu cho biến bằng cách sử dụng cú pháp như sau*:

CONST <Tên biến>: <Kiểu> = <Giá trị>;

Ví dụ:

```
CONST x:integer = 5;
```

Với khai báo biến x như trên, trong chương trình giá trị của biến x có thể thay đổi. (Điều này không đúng nếu chúng ta khai báo x là hằng).

IV. ĐỊNH NGHĨA KIỂU

- Ngoài các kiểu dữ liệu do Turbo Pascal cung cấp, ta có thể định nghĩa các kiểu dữ liệu mới dựa trên các kiểu dữ liệu đã có.
- Cú pháp:

TYPE <Tên kiểu> = <Mô tả kiểu>;
VAR <Tên biến>:<Tên kiểu>;

Ví dụ:

```
TYPE    Sothuc = Real;
        Tuoi = 1..100;
        ThuNgay = (Hai,Ba,Tu, Nam, Sau, Bay, CN)
VAR     x :Sothuc;
        tt : Tuoi;
        Day: ThuNgay;
```

V. BIỂU THỨC

Biểu thức (expression) là công thức tính toán mà trong đó bao gồm các phép toán, các hằng, các biến, các hàm và các dấu ngoặc đơn.

Ví dụ: (x +sin(y))/(5-2*x) biểu thức số học
(x+4)*2 = (8+y) biểu thức logic

Trong một biểu thức, thứ tự ưu tiên của các phép toán được liệt kê theo thứ tự sau:

- Lời gọi hàm.
- Dấu ngoặc ()
- Phép toán một ngôi (NOT, -).
- Phép toán *, /, DIV, MOD, AND.
- Phép toán +, -, OR, XOR
- Phép toán so sánh =, <, >, <=, >=, <>, IN

VI. CÂU LỆNH

6.1. Câu lệnh đơn giản

- ***Câu lệnh gán (:=)***: <Tên biến>**:=**<Biểu thức>;
- Các lệnh xuất nhập dữ liệu: **READ/READLN, WRITE/WRITELN.**
- Lời gọi hàm, thủ tục.

6.2. Câu lệnh có cấu trúc

- Câu lệnh ghép: **BEGIN ... END;**
- Các cấu trúc điều khiển: **IF.., CASE..., FOR..., REPEAT..., WHILE...**

6.3. Các lệnh xuất nhập dữ liệu

6.3.1. Lệnh xuất dữ liệu

Để xuất dữ liệu ra màn hình, ta sử dụng ba dạng sau:

(1)WRITE(<tham số 1> [, <tham số 2>,...]);

(2)WRITELN(<tham số 1> [, <tham số 2>,...]);

(3)WRITELN;

Các thủ tục trên có chức năng như sau:

(1) Sau khi xuất giá trị của các tham số ra màn hình thì con trỏ không xuống dòng.

(2) Sau khi xuất giá trị của các tham số ra màn hình thì con trỏ xuống đầu dòng tiếp theo.

(3) Xuất ra màn hình một dòng trống.

Các tham số có thể là các hằng, biến, biểu thức. Nếu có nhiều tham số trong câu lệnh thì các tham số phải được phân cách nhau bởi dấu phẩy.

Khi sử dụng lệnh WRITE/WRITELN, ta có hai cách viết: ***không qui cách*** và ***có qui cách***:

- ***Viết không qui cách***: dữ liệu xuất ra sẽ được canh lề ở phía bên trái. Nếu dữ liệu là số thực thì sẽ được in ra dưới dạng biểu diễn khoa học.

Ví dụ:

```
WRITELN(x); WRITE(sin(3*x));
```

- ***Viết có qui cách***: dữ liệu xuất ra sẽ được canh lề ở phía bên phải.

Ví dụ:

```
WRITELN(x:5); WRITE(sin(13*x):5:2);
```

Câu lệnh	Kết quả trên màn hình
Writeln('Hello');	Hello
Writeln('Hello':10);	Hello
Writeln(500);	500
Writeln(500:5);	500
Writeln(123.457)	1.2345700000E+02
Writeln(123.45:8:2)	123.46

6.3.2. Nhập dữ liệu

Để nhập dữ liệu từ bàn phím vào các biến có kiểu dữ liệu chuẩn (trừ các biến kiểu BOOLEAN), ta sử dụng cú pháp sau đây:

READLN(<biến 1> [,<biến 2>,...,<biến n>]);

Chú ý: *Khi gặp câu lệnh* **READLN;** *(không có tham số), chương trình sẽ dừng lại chờ người sử dụng nhấn phím ENTER mới chạy tiếp.*

6.4. Các hàm và thủ tục thường dùng trong nhập xuất dữ liệu

- Hàm **KEYPRESSED**: Hàm trả về giá trị TRUE nếu như có một phím bất kỳ được nhấn, nếu không hàm cho giá trị là FALSE.
- Hàm **READKEY**: Hàm có chức năng đọc một ký tự từ bộ đệm bàn phím.

- Thủ tục **GOTOXY(X,Y:Integer)**: Di chuyển con trỏ đến cột X dòng Y.
- Thủ tục **CLRSCR**: Xoá màn hình và đưa con trỏ về góc trên bên trái màn hình.
- Thủ tục **CLREOL**: Xóa các ký tự từ vị trí con trỏ đến hết dòng.
- Thủ tục **DELLINE**: Xoá dòng tại vị trí con trỏ và dồn các dòng ở phía dưới lên.
- Thủ tục **TEXTCOLOR(color:Byte)**: Thiết lập màu cho các ký tự. Trong đó color [illegible]0,15].
- Thủ tục **TEXTBACKGROUND(color:Byte)**: Thiết lập màu nền cho màn hình.

BÀI TẬP MẪU

Bài tập 2.1: Viết chương trình nhập vào độ dài hai cạnh của tam giác và góc giữa hai cạnh đó, sau đó tính và in ra màn hình diện tích của tam giác.

Ý tưởng:

Công thức tính diện tích tam giác: S = $\frac{1}{2}a.b.\sin($[illegible] với a,b là độ dài 2 cạnh và [illegible]à góc kẹp giữa 2 cạnh a và b.

```
Program Tinh_dien_tich_tam_giac;
Var a,b,goc,dientich: Real;
Begin
   Write('Nhap vao do dai canh thu nhat: '); Readln(a);
   Write('Nhap vao do dai canh thu hai: '); Readln(b);
  Write('Nhap vao goc giua hai canh: '); Readln(goc);
   Dientich:=a*b*sin(goc)/2;
   Writeln('Dien tich cua tam giac la: ',Dientich:0:2);
   Readln;
End.
```

Bài tập 2.2: Viết chương trình tính $\sqrt[n]{x}$, x>0.

Ý tưởng:

Ta có: $\sqrt[n]{x}$ [illegible]

```
Program Tinh_can_bac_n_cua_x;
Var   x,S: Real;
      n: Word;
Begin
   Write('Nhap vao n= '); Readln(n);
   Write('Nhap vao x= '); Readln(x);
   S:=EXP(1/n*LN(x));
   Writeln('S = ',S:0:2);
   Readln;
End.
```

Bài tập 2.3: Viết chương trình nhập vào 2 số a, b. Sau đó hoán đổi giá trị của 2 số đó:

a/ Cho phép dùng biến trung gian.

```
Program Swap;
Var   a,b,tam: Integer;
Begin
   Write('Nhap vao a= '); Readln(a);
   Write('Nhap vao b= '); Readln(b);
```

```
    tam:=a;     {tam lấy giá trị của a}
    a:=b;       {a lấy giá trị của b}
    b:=tam; {b lấy lại giá trị của tam}
    Writeln('a = ',a,' b = ',b);
    Readln;
End.
```

b/ Không được phép dùng biến trung gian.

```
Program Swap;
Var   a,b: Integer;
Begin
    Write('Nhap vao a= '); Readln(a);
    Write('Nhap vao b= '); Readln(b);
    a:=a+b;     {a lấy tổng giá trị của a+b}
    b:=a-b; {b lấy giá trị của a}
    a:=a-b;     {a lấy lại giá trị của b}
    Writeln('a = ',a,' b = ',b);
    Readln;
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 2.4: Viết chương trình nhập vào các số nguyên: a, b, x, y, ... sau đó in ra màn hình kết quả của các biểu thức sau:

a/ $\frac{x\,y}{2\,\frac{x}{y}}$ b/ $\frac{(a\,4)(b\,2c\,3)}{\frac{r}{2h}\,9(a\,)}$ c/ x^y , x>0 d/ $e\sqrt{|a\,\sin^2(x)\,d|}$

Bài tập 2.5: Viết chương trình tính siện tích tam giác theo công thức sau:

$S = \sqrt{p(p\,a)(p\,b)(p\,c)}$ với $p = \frac{1}{2}(a+b+c)$

Bài tập 2.6: Viết chương trình tính khoảng cách từ một điểm $I(x_i,y_i)$ đến đường thẳng có phương trình D: Ax + By + C = 0.

Gợi ý:

Công thức tính khoảng cách: $h = \frac{A.x_i\,B.y_i\,C}{\sqrt{A^2\,B^2}}$

Bài tập 2.7: Viết chương trình tách một số n thành 2 số a, b sao cho tích $P=a*b^2$ đạt cực đại với n được nhập vào từ bàn phím.

Gợi ý:

Gọi x là số thứ hai thì số thứ nhất là: (n-x). Theo đề ta có: $P(x) = x^2.(n-x)$.

Hàm P đạt cực đại khi $P'(x) = -3x^2 + 2nx = 0$ ➔ $x = 2n/3$.

Bài tập 2.8: Màn hình đồ họa của một máy tính có độ phân giải: 640x480. Biết rằng, mỗi điểm trên màn hình chiếm 1 byte. Hỏi cần bao nhiêu byte để lưu trữ toàn bộ màn hình đồ họa đó?

Có 2 sinh viên viết chương trình tính số byte lưu trữ màn hình đồ họa:

```
Program Sinhvien1;
Var   a,b:integer;
```

```
        s:Word;
Begin
   a:=640; b:=480;
   s:=a*b;
   writeln(s);  readln;
End.

Program Sinhvien2;
Var   a,b:Word;
        s: LongInt;
Begin
   a:=640; b:=480;
   s:=a*b;
   writeln(s);  readln;
End.
```

Hãy cho biết 2 chương trình trên cho kết quả đúng hay sai? Tại sao?

Bài tập 2.9: Màn hình đồ họa của một máy tính có độ phân giải: 640x480. Biết rằng, mỗi điểm trên màn hình chiếm 1 byte. Hỏi cần bao nhiêu byte để lưu trữ một vùng có kích thước bằng 1/10 màn hình đồ họa đó?

Có 2 sinh viên viết chương trình giải bài toán này như sau:

```
Program Sinhvien1;
Var   a,b:Word;
        s: LongInt;
Begin
   a:=640; b:=480;
   s:=a;
   s:=s*b;
   s:=s DIV 10;
   writeln(s);  readln;
End.

Program Sinhvien2;
Var   a,b:Word;
        s: LongInt;
Begin
   a:=640; b:=480;
   s:=a*b DIV 10;
   writeln(s);  readln;
End.
```

Hãy cho biết 2 chương trình trên cho kết quả đúng hay sai? Tại sao?

Chương 3
CÁC CÂU LỆNH CÓ CẤU TRÚC

I. CÂU LỆNH RẼ NHÁNH

1.1. Lệnh IF Cú pháp:

(1)IF B THEN S;

(2)IF B THEN S1 ELSE S2;

Chú ý: *Khi sử dụng câu lệnh IF thì đứng trước từ khoá ELSE không được có dấu chấm phẩy (;).*

1.2. Lệnh CASE Cú pháp:

Dạng 1	Dạng 2
CASE B OF	**CASE B OF**
Const 1: S_1;	**Const 1: S_1;**
Const 2: S_2;	**Const 2: S_2;**
...	**...**
Const n: S_n;	**Const n: S_n;**
END;	**ELSE S_{n+1};**
	END;

Trong đó:

- ☞ B: Biểu thức kiểu vô hướng đếm được như kiểu nguyên, kiểu logic, kiểu ký tự, kiểu liệt kê.
- ☞ Const i: Hằng thứ i, có thể là một giá trị hằng, các giá trị hằng (phân cách nhau bởi dấu phẩy) hoặc các đoạn hằng (dùng hai dấu chấm để phân cách giữa giá trị đầu và giá trị cuối).
- ☞ Giá trị của biểu thức và giá trị của tập hằng i (i=1¸n) phải có cùng kiểu.

Khi gặp lệnh CASE, chương trình sẽ kiểm tra:

- Nếu giá trị của biểu thức B nằm trong tập hằng const i thì máy sẽ thực hiện lệnh S_i tương ứng.
- Ngược lại:+ Đối với dạng 1: Không làm gì cả.

+ Đối với dạng 2: thực hiện lệnh S_{n+1}.

II. CÂU LỆNH LẶP

2.1. Vòng lặp xác địnhCó hai dạng sau:

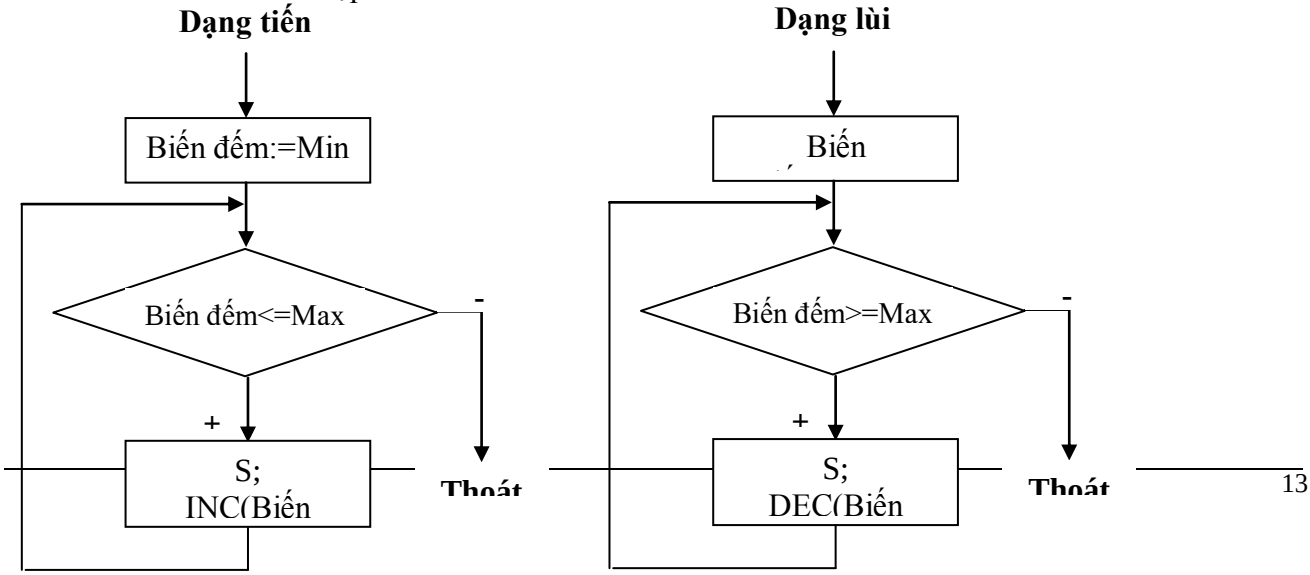
❶ ***Dạng tiến***

FOR <biến đếm>:=<giá trị Min> TO <giá trị Max> DO S;

❷ ***Dạng lùi***

FOR <biến đếm>:=<giá trị Max> DOWNTO <giá trị Min> DO S;

Sơ đồ thực hiện vòng lặp FOR:



Chú ý: Khi sử dụng câu lệnh lặp FOR cần chú ý các điểm sau:

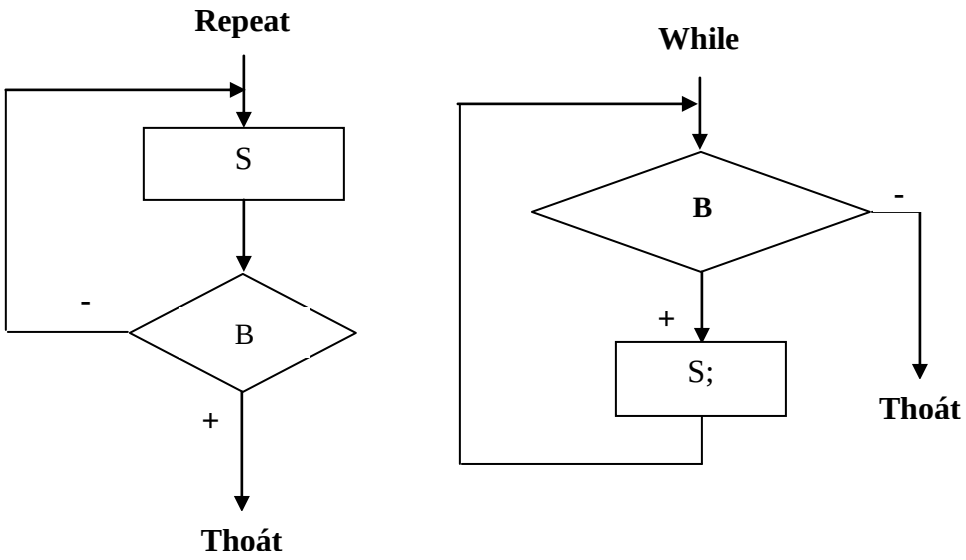
- *Không nên tuỳ tiện thay đổi giá trị của biến đếm bên trong vòng lặp FOR vì làm như vậy có thể sẽ không kiểm soát được biến đếm.*
- *Giá trị Max và Min trong câu lệnh FOR sẽ được xác định ngay khi vào đầu vòng lặp. Do đó cho dù trong vòng lặp ta có thay đổi giá trị của nó thì số lần lặp cũng không thay đổi.*

5.3.2. Vòng lặp không xác định

Dạng REPEAT	Dạng WHILE
Repeat **S;** **Until B;**	**While B Do S;**

Ý nghĩa:

- ***Dạng REPEAT***: Lặp lại công việc S cho đến khi biểu thức B=TRUE thì dừng.
- ***Dạng WHILE***: Trong khi biểu thức B=TRUE thì tiếp tục thực hiện công việc S.



BÀI TẬP MẪU

Bài tập 3.1: Viết chương trình nhập vào một số nguyên và kiểm tra xem số vừa nhập là số chẵn hay số lẻ.

```
Uses crt;
Var  x:integer;
Begin
  Write('Nhap vao mot so nguyen : ');  Readln(x);
  If x MOD 2=0 Then
    Writeln('So vua nhap vao la so chan')
  Else
    Writeln('So vua nhap vao la so le');
  Readln;
End.
```

Bài tập 3.2: Viết chương trình giải phương trình bậc nhất ax+b=0

```
Uses Crt;
Var  a,b,x : real;
Begin
  Write('a = '); Readln(a);
  Write('b = '); Readln(b);
  If a = 0 Then      { Nếu a bằng 0 }
    If b = 0 Then   { Trường hợp a = 0 và b = 0 }
      Writeln('Phuong trinh co vo so nghiem')
    Else    { Trường hợp a=0 và b [illegible] }
      Writeln('Phuong trinh vo nghiem')
  Else  { Trường hợp a [illegible] }
    Begin
      x:= -b/a;
      Writeln('Phuong trinh co nghiem la :',x:0:2);
    End;
  Readln;
End.
```

Bài tập 3.3: Viết chương trình nhập vào tuổi của một người và cho biết người đó là thiếu niên, thanh niên, trung niên hay lão niên. Biết rằng: nếu tuổi nhỏ hơn 18 là thiếu niên, từ 18 đến 39 là thanh niên, từ 40 đến 60 là trung niên và lớn hơn 60 là lão niên.

```
Uses crt;
Var tuoi:Byte;
Begin
  Write(Nhap vao tuoi cua mot nguoi:'); Readln(tuoi);
  Case tuoi Of
    1..17:  Writeln(Nguoi nay la thieu nien');
    18..39: Writeln(Nguoi nay la thanh nien');
    40..60: Writeln(Nguoi nay la trung nien');
    Else    Writeln(Nguoi nay la lao nien');
  End;
  Readln;
End.
```

Bài tập 3.4: Viết chương trình tính tổng S = 1+2+...+N

Cách 1: Dùng vòng lặp FOR.

```
Program TinhTong;
Uses crt;
Var N,i,S:integer;
Begin
  Clrscr;
  Write('Nhap vao gia tri cua N :'); Readln(N);
  S:=0;
```

```
   For i:=1 to N do S:=S+i;
   Writeln('Ket qua la :',S);
   Readln;
End.
```

Cách 2: Dùng vòng lặp REPEAT.

```
Program TinhTong;
Uses crt;
Var N,i,S:integer;
Begin
   Clrscr;
   Write('Nhap vao gia tri cua N :'); Readln(N);
   S:=0; i:=1;
   Repeat
      S:=S+i;
      i:=i+1;
   Until i>N;
   Writeln('Ket qua la :',S);
   Readln;
End.
```

Cách 3: Dùng vòng lặp WHILE.

```
Program TinhTong;
Uses crt;
Var N,i,S:integer;
Begin
   Clrscr;
   Write('Nhap vao gia tri cua N :'); Readln(N);
   S:=0; i:=1;
   While i<=N Do
      Begin
         S:=S+i;
         i:=i+1;
      End;
   Writeln('Ket qua la :',S);
   Readln;
End.
```

Bài tập 3.5: Viết chương trình nhập vào N số nguyên từ bàn phím. Hãy tính và in ra màn hình tổng của các số vừa được nhập vào.

Ý tưởng:

Dùng phương pháp cộng dồn. Cho vòng lặp FOR chạy từ 1 tới N, ứng với lần lặp thứ i, ta nhập vào số nguyên X và đồng thời cộng dồn X vào biến S.

```
Program  Tong;
Uses crt;
```

```
Var N,S,i,X : Integer;
Begin
   Clrscr; S:=0;
   For i:=1 To n Do
      Begin
         Write('Nhap so nguyen X= '); Readln(X);
         S:=S+X;
      End;
   Writeln('Tong cac so duoc nhap vao la: ',S);
   Readln;
End.
```

Bài tập 3.6: Viết chương trình nhập vào các số nguyên cho đến khi nào gặp số 0 thì kết thúc. Hãy đếm xem có bao nhiêu số chẵn vừa được nhập vào.

Ý tưởng:

Bài toán này không biết chính xác số lần lặp nên ta không thể dùng vòng lặp FOR. Vì phải nhập vào số nguyên N trước, sau đó mới kiểm tra xem N=0? Do đó ta nên dùng vòng lặp REPEAT.

```
Program  Nhapso;
Uses crt;
Var N,dem : Integer;
Begin
   Clrscr; dem:=0;
   Repeat
      Write('Nhap vao mot so nguyen N= '); Readln(N);
      If N MOD 2 = 0 Then dem:=dem+1;
   Until N=0;
   Writeln('Cac so chan duoc nhap vao la: ',dem);
   Readln;
End.
```

Bài tập 3.7: Viết chương trình tính số Pi với độ chính xác Epsilon, biết:

Pi/4 = 1-1/3+1/5-1/7+...

Ý tưởng:

Ta thấy rằng, mẫu số là các số lẻ có qui luật: 2*i+1 với i=1,...,n. Do đó ta dùng i làm biến chạy.

Vì tính số Pi với độ chính xác **Epsilon** nên không biết trước được cụ thể số lần lặp, do đó ta phải dùng vòng lặp WHILE hoặc REPEAT. Có nghĩa là phải lặp cho tới khi t=4/(2*i+1) Epsilon thì dừng.

```
Uses Crt;
Const Epsilon=1E-4;
Var Pi,t:real;
   i,s:Integer;
Begin
   Pi:=4;  i:=1; s:=-1;
   t:=4/(2*i+1);
   While t>Epsilon Do
```

```
      Begin
         Pi:=Pi+s*t;
         s:=-s; i:=i+1;
         t:=4/(2*i+1);
      End;
   Writeln('So Pi = ',Pi:0:4);
   Readln;
End.
```

Bài tập 3.8: Viết chương trình nhập vào số nguyên N. In ra màn hình tất cả các ước số của N.

Ý tưởng:

Cho biến i chạy từ 1 tới N. Nếu N MOD i=0 thì viết i ra màn hình.

```
Uses Crt;
Var N,i : Integer;
Begin
   Clrscr;
   Write('Nhap so nguyen N= '); Readln(N);
   For i:=1 To N Do
      If N MOD i=0 Then    Write(i:5);
   Readln;
End.
```

Bài tập 3.9: Viết chương trình tìm USCLN và BSCNN của 2 số a, b được nhập vào từ bàn phím.

Ý tưởng:

- Tìm USCLN: Lấy số lớn trừ số nhỏ cho đến khi a=b thì dừng. Lúc đó: USCLN=a.
- BSCNN(a,b) = a*b DIV USCLN(a,b).

```
Uses crt;
Var a,b,aa,bb:integer;
Begin
   Write('Nhap a : '); Readln(a);
   Write('Nhap b : '); Readln(b);
   aa:=a; bb:=b;
   While aa<>bb Do
      Begin
         If aa>bb Then aa:=aa-bb Else    bb:=bb-aa;
      End;
   Writeln('USCLN= ',aa);
   Writeln('BSCNN= ',a*b DIV aa);
   Readln;
End.
```

Bài tập 3.10: Viết chương trình tìm các số có 3 chữ số $\overline{abc}$ sao cho: $\overline{abc} = a^3 + b^3 + c^3$.

Ý tưởng:

Dùng phương pháp vét cạn. Ta biết rằng: a có thể có giá trị từ 1 (vì a là số hàng trăm), b,c có thể có giá trị từ 0 Ta sẽ dùng 3 vòng lặp FOR lồng nhau để duyệt qua tất cả các trường hợp của a,b,c.

Ứng với mỗi bộ abc, ta sẽ kiểm tra: Nếu 100.a + 10.b + c = $a^3 + b^3 + c^3$ thì in ra bộ abc đó.

```
Uses crt;
Var a,b,c : Word;
Begin
   For a:=1 To 9 Do
      For b:=0 To 9 Do
         For c:=0 To 9 Do
            If (100*a + 10*b + c)=(a*a*a + b*b*b + c*c*c) Then Writeln(a,b,c);
   Readln;
End.
```

Bài tập 3.11: Viết chương trình nhập vào số tự nhiên N rồi thông báo lên màn hình số đó có phải là số nguyên tố hay không.

Ý tưởng:

N là số nguyên tố nếu N không có ước số nào từ 2 [redacted] div 2. Từ định nghĩa này ta đưa ra giải thuật:

- Đếm số ước số của N từ 2 [redacted] div 2 lưu vào biến d.
- Nếu d=0 thì N là số nguyên tố.

```
Uses crt;
Var N,i,d : Word;
Begin
   If N<2 Then Writeln(N,' khong phai la so nguyen to')
   Else
   Begin
      {Đếm số ước số}
      d:=0;
      For i:=2 To N div 2 Do
         If N MOD i=0 Then    d:=d+1;
      {Kiểm tra}
      If d=0 Then Writeln(N,' la so nguyen to')
      Else Writeln(N,' khong phai la so nguyen to');
   End;
   Readln;
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 3.12: Viết chương trình giải phương trình bậc hai: $ax^2 + bx + c = 0$, a[redacted].

Gợi ý:

- Tính Delta=b*b-4*a*c.
- Biện luận:

  Delta<0: Phương trình vô nghiệm.

  Delta=0: Phương trình có nghiệm kép: x = -b/(2*a).

  Delta>0: Phương trình có 2 nghiệm phân biệt: $x_{1,2}$ = (-b[redacted]QRT(Delta))/(2*a).

Bài tập 3.13: Viết chương trình nhập vào từ bàn phím: giờ, phút, giây. Cọng thêm một số giây cũng được nhập từ bàn phím. Hãy in ra kết quả sau khi cọng xong.

Gợi ý:

- Gọi số giây được cộng thêm là: ss. Gán giây:=giây+ss.
- Nếu giây[illegible]0 thì: phút:=phút + giây DIV 60 và giây:=giây MOD 60.
- Nếu phút[illegible]0 thì: giờ:=giờ + phút DIV 60 và phút:=phút MOD 60.

Bài tập 3.14: Viết chương trình tìm Max, Min của 4 số: a, b, c, d.

Bài tập 3.15: Viết chương trình nhập vào ngày, tháng, năm. Máy sẽ hiện lên ngày, tháng, năm hôm sau.

Gợi ý:

Biện luận theo tháng. Gom tháng thành 3 nhóm: tháng có 31 ngày (1,3,5,7,8,10,12), tháng có 30 ngày (4,6,9,11) và tháng 2 (có 28 hoặc 29 ngày tùy theo năm nhuận).

Dùng lệnh lựa chọn:

```
CASE thang OF
1,3,5,7,8,10,12: .....
4,6,9,11: .....
2: .....
END;
```

Bài tập 3.16: Viết chương trình in ra màn hình các giá trị của bảng mã ASCII từ 0[illegible]5.

Gợi ý:

Cho biến i chạy từ 0 [illegible]55. In ra màn hình i và CHR(i).

Bài tập 3.17: Viết chương trình in ra màn hình các số nguyên từ 1 đến 100 sao cho cứ 10 số thì xuống dòng.

Gợi ý:

Cho biến i chạy từ 1 [illegible]00. In ra màn hình i và kiểm tra: nếu i MOD 10=0 thì WRITELN.

Bài tập 3.18: Viết chương trình in ra màn hình bảng cữu chương.

Gợi ý:

Dùng 2 vòng lặp FOR lồng nhau: i là số bảng cữu chương (2...9), j là số thứ tự trong từng bảng cữu chương (1...10).

```
For i:=2 To 9 Do
   For j:=1 To 10 Do Writeln(i,'x',j,'=',i*j);
```

Bài tập 3.19: Viết chương trình tính các tổng sau:

S0 = n! = 1*2*...*n {***n giai thừa***}

S1 = 1 + 1/2 + ... + 1/n

S2 = 1 + 1/2! + ... + 1/n!

S3 = 1 + x + x2/2! + x3/3! + ... + xn/n!

S4 = 1 - x + x2/2! - x3/3! + ... + $(-1)^n x^n/n!$

S5 = 1 + sin(x) + $\sin^2(x)$ + ... + $\sin^n(x)$.

Bài tập 3.20: Viết chương trình để tìm lời giải cho bài toán sau:

Trong giỏ vừa thỏ vừa gà,
Một trăm cái cẳng bốn ba cái đầu.

Hỏi có mấy gà mấy thỏ?

Bài tập 3.21: Viết chương trình nhập vào một số nguyên dương. Hãy thông báo lên màn hình số đó có bao nhiêu chữ số và tổng các chữ số của số đó.

Gợi ý:

Dùng vòng lặp WHILE. Trong khi N>0 thì: lấy ra chữ số cuối cùng của N để tính bằng phép toán MOD 10, sau đó bỏ bớt đi chữ số cuối cùng của N bằng phép toán DIV 10.

Bài tập 3.22: Viết chương trình in ra màn hình tất cả các số nguyên tố từ 2 đến N. Với N được nhập từ bàn phím.

Bài tập 3.23: Viết chương trình phân tích một số ra thừa số nguyên tố. Ví dụ: N=100 sẽ in ra màn hình:

```
100 | 2
 50 | 2
 25 | 5
  5 | 5
  1 |
```

Bài tập 3.24: Viết chương trình in ra các số nguyên từ 1 đến N^2 theo hình xoắn ốc với N được nhập vào từ bàn phím. Ví dụ, với N=5 ta có:

```
 1   2   3   4   5
16  17  18  19   6
15  24  25  20   7
14  23  22  21   8
13  12  11  10   9
```

Chương 4
CHƯƠNG TRÌNH CON: THỦ TỤC VÀ HÀM

I. KHÁI NIỆM VỀ CHƯƠNG TRÌNH CON

Chương trình con (CTC) là một đoạn chương trình thực hiện trọn vẹn hay một chức năng nào đó.

Trong Turbo Pascal, có 2 dạng CTC:

- Thủ tục (PROCEDURE): Dùng để thực hiện một hay nhiều nhiệm vụ nào đó.
- Hàm (FUNCTION): Trả về một giá trị nào đó (có kiểu vô hướng, kiểu string hoặc kiểu con trỏ). Hàm có thể sử dụng trong các biểu thức.

Ngoài ra, trong Pascal còn cho phép các CTC lồng vào nhau.

II. CẤU TRÚC CHUNG CỦA MỘT CHƯƠNG TRÌNH CÓ SỬ DỤNG CTC

```
PROGRAM  Tên_chương_trình;
USES CRT;
CONST  .....;
TYPE     .....;
VAR      .....;
PROCEDURE  THUTUC[(Các tham số)];
[Khai báo Const, Type, Var]
BEGIN
   .....
END;
FUNCTION  HAM[(Các tham số)]:<Kiểu dữ liệu>;
[Khai báo Const, Type, Var]
BEGIN
   .....
   HAM:=<Giá trị>;
END;

BEGIN {Chương trình chính}
   .....
   THUTUC[(...)];
   .....
   A:= HAM[(...)];
   .....
END.
```

Chú ý: Trong quá trình xây dựng CTC, khi nào thì nên dùng thủ tục/hàm?

Dùng hàm	Dùng thủ tục
- Kết quả của bài toán trả về ***1 giá trị duy nhất (kiểu vô hướng, kiểu string hoặc kiểu con trỏ).*** - Lời gọi CTC cần nằm trong các biểu thức tính toán.	- Kết quả của bài toán ***không trả về giá trị nào*** hoặc trả về ***nhiều giá trị*** hoặc ***trả về kiểu dữ liệu có cấu trúc (Array, Record, File).*** - Lời gọi CTC không nằm trong các biểu thức tính toán.

Ví dụ 1: Viết CTC để tính n! = 1.2...n.

Ý tưởng: Vì bài toán này trả về 1 giá trị duy nhất nên ta dùng hàm.

```
Function GiaiThua(n:Word):Word;
```

```
Var P, i:Word;
Begin
   P:=1;
   For i:=1 To n Do P:=P*i;
   GiaiThua:=P;
End;
```

Ví dụ 2: Viết chương trình con để tìm điểm đối xứng của điểm (x,y) qua gốc tọa độ.
Ý tưởng: Vì bài toán này trả về tọa độ điểm đối xứng (xx,yy) gồm 2 giá trị nên ta dùng thủ tục.

```
Procedure DoiXung(x,y:Integer; Var xx,yy:Integer);
Begin
   xx:=-x;
   yy:=-y;
End;
```

CHÚ Ý: Trong 2 ví dụ trên:

- **n, x, y** được gọi là **tham trị** (*không có từ khóa* **var** *đứng trước*) vì sau khi ra khỏi CTC giá trị của nó ***không bị thay đổi.***
- **xx, yy** được gọi là **tham biến** (*có từ khóa* **var** *đứng trước*) vì sau khi ra khỏi CTC giá trị của nó ***bị thay đổi.***

III. BIẾN TOÀN CỤC VÀ BIẾN ĐỊA PHƯƠNG

- **Biến toàn cục**: là các biến được khai báo trong chương trình chính. Các biến này có tác dụng ở mọi nơi trong toàn bộ chương trình.
- **Biến địa phương:** là các biến được khai báo trong các CTC. Các biến này chỉ có tác dụng trong phạm vi CTC đó mà thôi.

Chú ý: Trong một CTC, nếu biến toàn cục trùng tên với biến địa phương thì biến địa phương được ưu tiên hơn.
Ví dụ:

```
Program  KhaoSatBien;
Var a,b: Integer;    {biến toàn cục}
Procedure  ThuBien;
Var  a: Integer;  {biến địa phương}
Begin
   a:=10;
   Writeln('A=',a,'B=',b);
End;
Begin
   a:=50;
   b:=200;
   ThuBien;                    {A=10  B=200}
   Writeln('A=',a,'B=',b);    {A=50  B=200}
End.
```

IV. ĐỆ QUI

4.1. Khái niệm đệ qui

Trong một chương trình, một CTC có thể gọi một CTC khác vào làm việc. Nếu như CTC đó ***gọi lại chính nó*** thì gọi là sự đệ qui.

4.2. Phương pháp thiết kế giải thuật đệ qui

- Tham số hóa bài toán
- Tìm trường hợp suy biến.
- Phân tích các trường hợp chung (đưa về các bài toán cùng loại nhưng nhỏ hơn).

Ví dụ: Viết hàm đệ qui để tính n! = 1.2...n.

- Tham số hóa: n! = Factorial(n);
- Factorial(0) = 1 (trường hợp suy biến)
- Factorial(n) = n*Factorial(n-1) (trường hợp chung)

```
Function  Factorial(N:integer):Longint;
Begin
   If N=0 Then  Factorial:=1
   Else Factorial:=N*factorial(N-1);   { lời gọi đệ qui }
End;
```

4.3. Giải thuật quay lui

Bài toán:

Hãy xây dựng các bộ giá trị gồm n thành phần (x_1,...,x_n) từ một tập hữu hạn cho trước sao cho các bộ đó thỏa mãn yêu cầu B cho trước nào đó.

Phương pháp chung

Giả sử đã xác định được k-1 phần tử đầu tiên của dãy: x_1,...,x_{k-1}. Ta cần xác định phần tử thứ k. Phần tử này được xác định theo cách sau:

- Giả sử T_k: tập tất cả các giá trị mà phần tử x_k có thể nhận được. Vì tập T_k hữu hạn nên ta có thể đặt n_k là số phần tử của T_k theo một thứ tự nào đó, tức là ta có thể thành lập một ánh xạ 1-1 từ tập T_k lên tập {1, 2, ..., n_k}.
- Xét j[illegible]1, 2, ..., n_k}. Ta nói rằng "***j chấp nhận được***" nếu ta có thể bổ sung phần tử thứ j trong T_k với tư cách là phần tử x_k vào trong dãy x_1,...,x_{k-1} để được dãy x_1,...,x_k.
- Nếu k=n: Bộ (x_1,...,x_k) thỏa mãn yêu cầu B, do đó bộ này được thu nhận.
- Nếu k<n: Ta thực hiện tiếp quá trình trên, tức là phải bổ sung tiếp các phần tử x_{k+1} vào dãy x_1,...,x_k.

Sau đây là thủ tục đệ qui cho giải thuật quay lui:

```
Procedure THU(k:Integer);
Var j:Integer;
Begin
   For j:=1 To nk Do
      If <j chấp nhận được> Then
         Begin
            <Xác định xk theo j>;
            If k=n Then  <Ghi nhận một bộ giá trị>
            Else  THU(k+1);  {Quay lui}
         End;
End;
```

Ví dụ: Liệt kê các dãy nhị phân có độ dài n.

```
Program DayNhiPhan;
Var   b:Array[1..20] Of 0..1;    {Dãy nhị phân có độ dài tối đa là 20}
```

```
      n:Byte;
Procedure InKetQua;
Var i:Byte;
Begin
   For i:=1 To n Do  Write(b[i]);
   Writeln;
End;
Procedure THU(k:Byte);
Var j:Byte;
Begin
   For j:=0 To 1 Do       {Tập giá trị của dãy nhị phân}
      Begin
         b[k]:= j;
         If k=n Then  InKetQua
         Else  THU(k+1);  {Quay lui}
      End;
End;
Begin
   Write('n = '); Readln(n);
   THU(1);
   Readln;
End.
```

V. TẠO THƯ VIỆN (UNIT)

5.1. Cấu trúc của một Unit

```
UNIT  <Tên Unit>;    {phải trùng với tên file}
INTERFACE
   USES .....;
   CONST.....;
   TYPE .....;
   VAR  .....;
   Procedure  <Tên thủ tục>[(Các tham số)];
   Function  <Tên hàm>[(Các tham số)]:<Kiểu hàm>;
IMPLEMENTATION
   Procedure  <Tên thủ tục>[(Các tham số)];
   [Các khai báo]
   Begin
      .....
   End;

   Function  <Tên hàm>[(Các tham số)]:<Kiểu hàm>;
   [Các khai báo]
   Begin
```

```
    .............
  End;
END.
```

Chú ý:

- Tên của Unit phải trùng với tên file.
- Chỉ có những chương trình con được khai báo ở phần INTERFACE mới sử dụng được ở các chương trình khác.
- Các thủ tục và hàm được khai báo ở phần INTERFACE thì bắt buộc phải có trong phần IMPLEMENTATION.

5.2. Ví dụ minh họa

Tạo Unit MYTOOL lưu ở file MYTOOL.PAS.

```
UNIT  MYTOOL;
INTERFACE
  USES CRT;
  VAR  m:Integer;
  Procedure  WriteXY(x,y:Integer; St:String);
  Function  UCLN(a,b:Integer):Integer;
  Function  NGUYENTO(n:Word):Word;
IMPLEMENTATION
  Procedure  WriteXY(x,y:Integer; St:String);
  Var i:Byte;
  Begin
    Gotoxy(x,y); Write(St);
  End;
  Function  UCLN(a,b:Integer):Integer;
  Begin
    While a<>b Do
      Begin
        If a>b Then a:=a-b  Else  b:=b-a;
      End;
    UCLN:=a;
  End;
  Function  NGUYENTO(n:Word):Boolean;
  Var d,i:Word;
  Begin
    d:=0;
    For i:=2 To n DIV 2 Do
      If n MOD i=0 Then d:=d+1;
    NGUYENTO:=d=0;
  End;
END.
```

Bây giờ, ta có thể viết một chương trình có sử dụng Unit MYTOOL.

```
Uses Crt, MyTool;
Var  a,b:Integer;
Begin
   CLRSCR;
   Write(10,5,'CHUONG TRINH MINH HOA');
   Write('Nhap a = '); Readln(a);
   Write('Nhap b = '); Readln(b);
   Writeln('UCLN cua ',a,' va ',b,' la:',UCLN(a,b));
   Write('Nhap m = '); Readln(m);
   If  NGUYENTO(m) Then
      Writeln(m,' la so nguyen to!')
   Else
      Writeln(m,' khong phai la so nguyen to!')
   Readln;
End.
```

BÀI TẬP MẪU

Bài tập 4.1: Viết hàm tìm Max của 2 số thực x,y.

```
Var a,b:Real;
Function Max(x,y:Real):Real;
Begin
   If x>y Then Max:=x Else Max:=y;
End;
Begin
   Write('Nhap a='); Readln(a);
   Write('Nhap b='); Readln(b);
   Writeln('So lon nhat trong 2 so la: ', Max(a,b));
   Readln;
End.
```

Bài tập 4.2: Viết hàm LOWCASE(c:char):char; để đổi chữ cái hoa c thành chữ thường.

Ý tưởng:

Trong bảng mã ASCII, số thứ tự của chữ cái hoa nhỏ hơn số thứ tự của chữ cái thường là 32. Vì vậy ta có thể dùng 2 hàm CHR và ORD để chuyển đổi.

```
Uses crt;
Var ch:Char;
Function LOWCASE(c:Char):Char;
Begin
   If c IN ['A'..'Z'] Then LOWCASE:=CHR(ORD(c)+32)
   Else LOWCASE:=c;
End;
Begin
```

```
   Write('Nhap ký tu ch='); Readln(ch);
   Writeln('Ky tu hoa la: ', LOWCASE(ch));
   Readln;
End.
```

Bài tập 4.3: Viết thủ tục để hoán đổi hai gía trị x,y cho nhau.

```
Var a,b:Real;
Function Swap(Var x,y:Real);
Var Tam:Real;
Begin
   Tam:=x; x:=y; y:=Tam;
End;
Begin
   Write('Nhap a='); Readln(a);
   Write('Nhap b='); Readln(b);
   Swap(a,b);
   Writeln('Cac so sau khi hoan doi:  a=', a:0:2,' b=',b:0:2);
   Readln;
End.
```

Bài tập 4.4: Viết hàm XMU(x:Real;n:Byte):Real; để tính giá trị x^n.

```
Var   x:Real;
      n:Byte;
Function XMU(x:Real;n:Byte):Real;
Var i:Byte; S:Rea;
Begin
   S:=1;
   For i:=1 To n Do S:=S*x;
   XMU:=S;
End;
Begin
   Write('Nhap x='); Readln(x);
   Write('Nhap n='); Readln(n);
   Writeln('x mu n = ', XMU(x,n):0:2);
   Readln;
End.
```

Bài tập 4.5: Viết thủ tục KHUNG(x1,y1,x2,y2:Integer); để vẽ một khung hình chữ nhật có đỉnh trên bên trái là (x1,y1) và đỉnh dưới bên phải là (x2,y2).

Ý tưởng:

Dùng các ký tự mở rộng trong bảng mã ASCII: (#179), (196), (#218), (#192), (#191), (#217).

```
Uses crt;
Procedure Khung(x1,y1,x2,y2:Integer);
Var i,j:Integer;
Begin
  Gotoxy(x1,y1); Write(#218); {Vẽ
  Gotoxy(x1,y2); Write(#192); {Vẽ
 {Vẽ 2 viền ngang của khung}
  For i:=x1+1 To x2-1 do
   Begin
    Gotoxy(i,y1); Write(#196);
    Gotoxy(i,y2); Write(#196);
   End;
Gotoxy(x2,y1); Write(#191); {Vẽ
  Gotoxy(x2,y2); Write(#217); {Vẽ
 {Vẽ 2 viền dọc của khung}
  For j:=y1+1 To y2-1 do
   Begin
    Gotoxy(x1,j); Write(#179);
    Gotoxy(x2,j); Write(#179);
   End;
End;
Begin
  Clrscr;
  Khung(10,5,40,20);
  Readln;
End.
```

Bài tập 4.6: Viết thủ tục PHANTICH(n:Integer); để phân tích số nguyên n ra thừa số nguyên tố.

```
Uses crt;
Var n:Integer;
Procedure PHANTICH(n:Integer);
Var i:Integer;
Begin
  i:=2;
  While n<>1 Do
   Begin
     While n MOD i=0 Do
      Begin
        Writeln(n:5,'|',i:2);
        n:=n Div i;
      End;
     i:=i+1;
   End;
  Writeln(n:5,'|');
End;
Begin
  Write('Nhap n='); Readln(n);
  PHANTICH(n);
  Readln;
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 4.7: Viết 2 hàm tìm Max , min của 3 số thực.

Bài tập 4.8: Viết các hàm đệ quy để tính:

$S_1 = 1+2+3+.....+n$;

$S_2 = 1+1/2 + ...+ 1/n$;

$S_3 = 1-1/2 +.....+ (-1)^{n+1}\ 1/n$

$S_4 = 1 + \sin(x) + \sin^2(x) + ...+ \sin^n(x)$

Bài tập 4.9: Viết hàm đệ quy để tính C^k_n biết :

$C^n_n = 1$, $C^0_n = 1$, $C^k_n = C^{k-1}_{n-1} + C^k_{n-1}$.

Bài tập 4.10: Cho m , n nguyên dương . Lập hàm đệ quy tính:

$$A(m,n) = \begin{cases} n+1 & , \ m = 0 \\ A(m-1,1) & , \ n = 0 \\ A(m-1, A(m,n-1)) & , \ m > 0 \text{ và } n > 0 \end{cases}$$

Bài tập 4.11: Lập hàm đệ qui để tính dãy Fibonacci:

$$F(n) = \begin{cases} 1 & , \ n = 1 \text{ hoặc } n = 2 \\ F(n-1) + F(n-2) & , \ n > 2 \end{cases}$$

Bài tập 4.12: Viết hàm đệ qui tìm USCLN của 2 số.

Bài tập 4.13: Viết thủ tục để in ra màn hình số đảo ngược của một số nguyên cho trước theo 2 cách: đệ qui và không đệ qui.

Bài tập 4.14: Viết chương trình in ra màn hình các hoán vị của n số nguyên đầu tiên.

Bài tập 4.15: Xây dựng một Unit SOHOC.PAS chứa các thủ tục và hàm thực hiện các chức năng sau:

- Giải phương trình bặc nhất.
- Giải phương trình bặc hai.
- Tìm Max/Min của 2 số a,b.
- Tìm USCLN và BSCNN của 2 số nguyên a,b.
- Kiểm tra số nguyên dương n có phải là số nguyên tố hay không?
- Kiểm tra số nguyên dương n có phải là số hoàn thiện hay không?
- Đổi một số nguyên dương n sang dạng nhị phân.
- In ra màn hình bảng cữu chương từ 2 [redacted]

Sau đó, tự viết các chương trình có sử dụng Unit SOHOC vừa được xây dựng ở trên.

Chương 5
DỮ LIỆU KIỂU MẢNG (ARRAY)

I. KHAI BÁO MẢNG

Cú pháp:

TYPE <Kiểu mảng> = ARRAY [chỉ số] OF <Kiểu dữ liệu>;

VAR <Biến mảng>:<Kiểu mảng>;

hoặc khai báo trực tiếp:

VAR <Biến mảng> : ARRAY [chỉ số] OF <Kiểu dữ liệu>;

Ví dụ:

```
TYPE   Mangnguyen = Array[1..100] of Integer;
       Matrix = Array[1..10,1..10] of Integer;
       MangKytu = Array[Byte] of Char;
VAR    A: Mangnguyen;
       M: Matrix;
       C: MangKytu;
```

hoặc:

```
VAR    A: Array[1..100] of Integer;
       C: Array[Byte] of Char;
```

II. XUẤT NHẬP TRÊN DỮ LIỆU KIỂU MẢNG

- Để truy cập đến phần tử thứ k trong mảng một chiều A, ta sử dụng cú pháp: A[k].
- Để truy cập đến phần tử (i,j) trong mảng hai chiều M, ta sử dụng cú pháp: M[i,j].
- Có thể sử dụng các thủ tục READ(LN)/WRITE(LN) đối với các phần tử của biến kiểu mảng.

BÀI TẬP MẪU

Bài tập 5.1: Viết chương trình tìm giá trị lớn nhất của một mảng chứa các số nguyên gồm N phần tử.

Ý tưởng:

- Cho số lớn nhất là số đầu tiên: Max:=a[1].
- Duyệt qua các phần tử a[i], với i chạy từ 2 tới N: Nếu a[i]>Max thì thay Max:=a[i];

```
Uses Crt;
Type Mang = ARRAY[1..50] Of Integer;
Var  A:Mang;
     N,i,Max:Integer;
Begin
  {Nhập mảng}
  Write('Nhap N='); Readln(N);
  For i:=1 To N Do
     Begin
        Write('A[',i,']='); Readln(A[i]);
     End;
  {Tìm phần tử lớn nhất}
  Max:=A[1];
  For i:=2 To N Do
     If Max<A[i] Then Max:=A[i];
```

```
   {In kết quả ra màn hình}
   Writeln('Phan tu lon nhat cua mang: ', Max);
   Readln;
End.
```

Bài tập 5.2: Viết chương trình tính tổng bình phương của các số âm trong một mảng gồm N phần tử.

Ý tưởng:

Duyệt qua tất cả các phần tử A[i] trong mảng: Nếu A[i]<0 thì cộng dồn $(A[i])^2$ vào biến S.

```
Uses Crt;
Type Mang = ARRAY[1..50] Of Integer;
Var   A:Mang;
      N,i,S:Integer;
Begin
   {Nhập mảng}
   Write('Nhap N='); Readln(N);
   For i:=1 To N Do
      Begin
         Write('A[',i,']='); Readln(A[i]);
      End;
   {Tính tổng}
   S:=0;
   For i:=1 To N Do
      If A[i]<0 Then S:=S+A[i]*A[i];
   {In kết quả ra màn hình}
   Writeln('S= ', S);
   Readln;
End.
```

Bài tập 5.3: Viết chương trình nhập vào một mảng gồm N số nguyên. Sắp xếp lại mảng theo thứ tự tăng dần và in kết quả ra màn hình.

Ý tưởng:

Cho biến i chạy từ 1 đến N-1, đồng thời cho biến j chạy từ i+1 đến N: Nếu A[i]>A[j] thì đổi chỗ A[i], A[j].

```
Uses Crt;
Type Mang = ARRAY[1..50] Of Integer;
Var   A:Mang;
      N,i,j,Tam:Integer;
Begin
   {Nhập mảng}
   Write('Nhap N='); Readln(N);
   For i:=1 To N Do
      Begin
         Write('A[',i,']='); Readln(A[i]);
      End;
```

```
   {Sắp xếp}
   For i:=1 To N-1 Do
      For j:=i+1 To N Do
         If A[i]>A[j] Then
            Begin
               Tam:=A[i]; A[i]:=A[j]; A[j]:=Tam;
            End;
   {In kết quả ra màn hình}
   Writeln('Ket qua sau khi sap xep:');
   For i:=1 To N Do Write(A[i]:5);
   Readln;
End.
```

Bài tập 5.4: Viết chương trình nhập vào một mảng A gồm N số nguyên và nhập thêm vào một số nguyên X. Hãy kiểm tra xem phần tử X có trong mảng A hay không?

Ý tưởng:

Dùng thuật toán tìm kiếm tuần tự. So sánh x với từng phần tử của mảng A. Thuật toán dừng lại khi x=A[i] hoặc i>N.

Nếu x=A[i] thì vị trí cần tìm là i, ngược lại thì kết quả tìm là 0 (không tìm thấy).

```
Uses Crt;
Type Mang = ARRAY[1..50] Of Integer;
Var   A:Mang;
      N,i,x:Integer;
Function TimKiem(x, N: Integer; A:Mang):Integer;
Var i:Integer;
   Begin
      I:=1;
      While (I <= N) and (X<>A[I]) do I:=I+1;
      If I <= N Then Timkiem:=I  Else Timkiem:=0;
   End;
Begin
   {Nhập mảng}
   Write('Nhap N='); Readln(N);
   For i:=1 To N Do
      Begin
         Write('A[',i,']='); Readln(A[i]);
      End;
   Write('Nhap X='); Readln(x);
   {Kết quả tìm kiếm}
   If TimKiem(X,N,A)<>0 Then
      Writeln('Vi tri cua X trong mang la:', TimKiem(X,N,A))
   Else Writeln('X khong co trong mang.');
   Readln;
End.
```

Bài tập 5.5: Giả sử mảng A đã được sắp xếp theo thứ tự tăng dần. Viết hàm để kiểm tra xem phần tử X có trong mảng A hay không?

Ý tưởng:

So sánh x với phần tử ở giữa mảng A[giua]. Nếu x=A[giua] thì dừng (vị trí cần tìm là chỉ số của phần tử giữa của mảng). Ngược lại, nếu x>A[giua] thì tìm ở đoạn sau của mảng [giua+1,cuoi], ngược lại thì tìm ở đoạn đầu của mảng [dau,giua-1].

Sau đây là hàm cài đặt cho thuật toán này:

```
Function TimKiemNhiPhan(X, N: Integer; A: Mang):Integer;
Var   dau,cuoi,giua:Integer;
      Found:Boolean;
Begin
   dau:=1; {điểm mút trái của khoảng tìm kiếm}
   cuoi:=N; {điểm mút phải của khoảng tìm kiếm}
   Found:=False; {chưa tìm thấy}
   While (dau <=cuoi) and (Not Found) Do
      Begin
         giua:=(dau + cuoi) Div 2;
         If  X = A[giua] Then Found:=True {đã tìm thấy}
         Else
            If X > A[giua] Then dau:=giua+1
            Else cuoi:=giua-1;
      End;
   If Found Then TimKiemNhiPhan:= giua Else TimKiemNhiPhan:=0;
End;
```

Bài tập 5.6: Viết chương trình tìm ma trận chuyển vị của ma trận A.

Ý tưởng:

Dùng mảng 2 chiều để lưu trữ ma trận. Gọi B là ma trận chuyển vị của ma trận A, ta có: $B_{ij} = A_{ji}$.

```
Uses Crt;
Type Mang = ARRAY[1..10,1..10] Of Integer;
Var   A,B:Mang;
      m,n,i,j:Integer;
Begin
   {Nhập ma trận}
   Write('Nhap số dòng m='); Readln(m);
   Write('Nhap số cột n='); Readln(n);
   For i:=1 To m Do
      For j:=1 To n Do
         Begin
            Write('A[',i,j,']='); Readln(A[i,j]);
         End;
   {Tìm ma trận chuyển vị}
   For i:=1 To m Do
      For j:=1 To n Do    B[i,j]:=A[j,i];
   {In ma trận chuyển vị ra màn hình}
```

```
   For i:=1 To m Do
      Begin
         For j:=1 To n Do    Write(B[i,j]:5);
         Writeln;
      End;
   Readln;
End.
```

Bài tập 5.7: Cho một mảng 2 chiều A cấp mxn gồm các số nguyên và một số nguyên x. Viết chương trình thực hiện các công việc sau:

a/ Đếm số lần xuất hiện của x trong A và vị trí của chúng.

b/ Tính tổng các phần tử lớn nhất của mỗi dòng.

```
Uses Crt;
Type Mang = ARRAY[1..10,1..10] Of Integer;
Var   A:Mang;
      m,n,i,j,x,dem,S,max:Integer;
Begin
   {Nhập ma trận}
   Write('Nhap số dòng m='); Readln(m);
   Write('Nhap số cột n='); Readln(n);
   For i:=1 To m Do
      For j:=1 To n Do
         Begin
            Write('A[',i,j,']='); Readln(A[i,j]);
         End;
   {Nhập x}
   Write('Nhap x='); Readln(x);
   {Đếm số lãn xuất hiện của x và vị trí của x}
   dem:=0;
   Writeln('Vi tri cua x trong mang A: ');
   For i:=1 To m Do
      For j:=1 To n Do
         If x=A[i,j] Then
            Begin
               Write(i,j,' ; ');
               dem:=dem+1;
            End;
   Writeln('So lan xuat hien cua x trong mang A la: ',dem);
   {Tính tổng các phần tử lớn nhất của mỗi dòng}
   S:=0;
   For i:=1 To m Do {duyệt qua từng dòng}
      Begin
         {Tìm phần tử lớn nhất của dòng thứ i}
```

```
            Max:=A[i,1];
            For j:=2 To n Do    {duyệt từng phần tử của dòng thứ i}
               If max<A[i,j] Then max:=A[i,j];
            {Cộng max vào biến S}
            S:=S+max;
         End;
      Writeln('Tong cac phan tu lon nhat cua moi dong la: ',S);
      Readln;
End.
```

Bài tập 5.8: Giải phương trình bằng phương pháp chia nhị phân.

Ý tưởng:

Giả sử cần tìm nghiệm của phương trình f(x)=0 trên đoạn [a,b] với y=f(x) đồng biến và đơn trị trên đoạn [a,b]. Ta giải như sau:

Gọi m là trung điểm của đoạn [a,b]. Nếu f(m)*f(a)<0 thì giới hạn đoạn tìm nghiệm thành [a,m]. Tương tự đối với đoạn [m,b]. Quá trình này lặp lại cho đến khi f(m)< lức này ta có 1 nghiệm gần đúng là m.

Giả sử f(x) là một đa thức: $f(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$. Lúc này, ta có thể dùng mảng một chiều để lưu trữ các hệ số a_i của đa thức.

```
Uses Crt;
Type HESO=Array[0..20] Of Real;
Var   a:HESO;
      n:Byte;
      Min,Max,epsilon:Real;

Procedure NhapDaThuc;
Var i:Byte;
Begin
  Write('Bac cua da thuc: n= '); Readln(n);
  Writeln('Nhap cac he so cua da thuc:');
  For i:=0 To n Do
   Begin
    Write('a[',i,']='); Readln(a[i]);
   End;
  Writeln('Nhap doan tim nghiem:[a,b]');
  Write('a= '); Readln(Min);
  Write('b= '); Readln(Max);
  Write('Nhap sai so cua phuong trinh: '); Readln(epsilon);
End;
{Tính giá trị của đa thức}
Function f(x:Real):Real;
Var   S,tam:Real;
      i:Byte;
Begin
   S:=a[0]; tam:=1;
   For i:=1 To n Do
      Begin
         tam:=tam*x;
```

```
            S:=S+a[i]*tam;
        End;
    f:=S;
End;

Procedure TimNghiem(Min,Max:real);
Var m:Real;
Begin
  If f(Min)*f(Max)>0 Then Writeln('Phuong trinh vo nghiem.')
  Else If abs(f(Min))<epsilon Then Writeln('Nghiem la x=',min:0:2)
      Else If abs(f(Max))<epsilon Then Writeln('Nghiem la x=',max:0:2)
          Else
            Begin
              m:=(Min+Max)/2;
              If abs(f(m))<=epsilon Then Writeln('Nghiem la x=',m:0:2)
              Else If f(Min)*f(m)<0 Then TimNghiem(Min,m)
                   Else TimNghiem(m,Max);
            End;
End;
Begin
 NhapDaThuc;
 TimNghiem(Min,Max);
 Readln;
End.
```

Bài tập 5.9: Viết chương trình nhập vào số tự nhiên N (N lẻ), sau đó điền các số từ 1 đến n^2 vào trong một bảng vuông sao cho tổng các hàng ngang, hàng dọc và 2 đường chéo đều bằng nhau (bảng này được gọi là Ma phương).

Ví dụ: Với N=3 và N=5 ta có

2	7	6
9	5	1
4	3	8

Bắc

3	16	9	22	15
20	8	21	14	2
7	25	13	1	19
24	12	5	18	6
11	4	17	10	23

Tây Đông

Nam

Phương pháp:

Xuất phát từ ô bên phải của ô nằm giữa. Đi theo ***hướng đông bắc*** để điền các số 1, 2, ...

Khi điền số, cần chú ý một số nguyên tắc sau:

- Nếu vượt ra phía ngoài bên phải của bảng thì quay trở lại cột đầu tiên.
- Nếu vượt ra phía ngoài bên trên của bảng thì quay trở lại dòng cuối cùng.
- Nếu số đã điền k chia hết cho N thì số tiếp theo sẽ được viết trên cùng một hàng với k nhưng cách 1 ô về phía bên phải.

```
Uses Crt;
Var A:Array[1..20,1..20] Of Word;
    n,i,j,k:Word;
Begin
```

```
  Write('Nhap N= '); Readln(n);
  Clrscr;
 {Định vị ô xuất phát}
  i:=n DIV 2 + 1;
  j:=n DIV 2 + 2;
 {Điền các số k từ 1 đến n*n}
  For k:=1 To n*n Do
   Begin
     A[i,j]:=k;
     If k MOD n=0 Then j:=j+2
     Else Begin
                {Đi theo hướng đông bắc}
                 j:=j+1; i:=i-1;
          End;
     If j>n Then j:=j MOD n;
     If i=0 Then i:=n;
   End;
 {In kết quả ra màn hình}
  For i:=1 To n Do
   Begin
     For j:=1 To n Do write(a[i,j]:4);
     Writeln;
   End;
  Readln;
End.
```

Bài tập 5.10: Viết chương trình nhập vào 2 mảng số nguyên A, B đại diện cho 2 tập hợp (không thể có 2 phần tử trùng nhau trong một tập hợp). Trong quá trình nhập, phải kiểm tra: nếu phần tử vừa nhập vào đã có trong mảng thì không bổ sung vào mảng. In ra màn hình các phần tử là giao của 2 tập hợp A, B.

Ý tưởng:

Duyệt qua tất cả các phần tử a_i. Nếu a_i thì viết a_i ra màn hình.

```
Uses Crt;
Type Mang=ARRAY[1..50] Of Integer;

Var A,B:Mang;
   n,m:Byte;
Function KiemTra(x:Integer; n:Byte; A:Mang):Boolean;
Var i:Byte; Found:Boolean;
Begin
  Found:=False;
  i:=1;
  While (i<=n) AND (not Found) Do
    If x=A[i] Then Found:=True Else i:=i+1;
  KiemTra:=Found;
End;
Procedure NhapMang(Var n:Byte; Var A:Mang);
Var ch:Char;
   x:Integer;
Begin
```

```
  n:=0;
  Repeat
    Write('x='); Readln(x);
    If not KiemTra(x,n,A) Then
     Begin
       n:=n+1; A[n]:=x;
     End;
    Writeln('An ESC de ket thuc nhap!');
    ch:=Readkey;
  Until ch=#27;
End;
Procedure GiaoAB(n:Byte; A:Mang;m:Byte; B:Mang);
Var i:Byte;
Begin
  For i:=1 To n Do
   If KiemTra(A[i],m,B) Then Write(A[i]:4);
End;
Begin
  Clrscr;
  Writeln('Nhap mang A: ');
  NhapMang(n,A);
  Writeln('Nhap mang B: ');
  NhapMang(m,B);
  Writeln('Giao cua 2 mang A&B la: ');
  GiaoAB(n,A,m,B);
  Readln;
End.
```

Bài tập 5.11: Cho một mảng số nguyên gồm n phần tử. Tìm dãy con gồm m phần tử (m[redacted]) sao cho dãy con này có tổng lớn nhất. (Dãy con là dãy các phần tử liên tiếp nhau trong mảng).

```
Uses Crt;
Type Mang=ARRAY[1..50] Of Integer;
Var A:Mang;
    n,m,i,j,k:Byte;
    S,Max:Integer;
Begin
  Write('So phan tu cua mang: n= '); Readln(n);
  For i:=1 To n Do
   Begin
     Write('a[',i,']='); Readln(a[i]);
   End;
  Write('Nhap so phan tu cua day con: m= '); Readln(m);
 k:=1; {Vị trí phần tử đầu tiên của dãy con}
  {Giả sử m phần tử đầu tiên của mảng A là dãy con có tổng lớn nhất}
  Max:=0;
  For i:=1 To m Do Max:=Max+A[i];
{Tìm các dãy con khác}
  For i:=2 To n-m+1 Do
   Begin
    {Tính tổng của dãy con thứ i}
     S:=0;
```

```
    For j:=i To i+m-1 Do S:=S+A[j];
    If S>Max Then {Nếu dãy con tìm được có tổng lớn hơn dãy con trước}
      Begin
        Max:=S; {Thay tổng mới}
        k:=i;       {Thay vị trí đầu tiên của dãy con mới}
      End;
  End;
 Writeln('Day con co tong lon nhat la:');
  For i:=k To k+m-1 Do Write(A[i]:5);
  Readln;
End.
```

Bài tập 5.12: Viết chương trình in ra màn hình tam giác Pascal. Ví dụ, với n=4 sẽ in ra hình sau:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Ý tưởng:

Tam giác Pascal được tạo ra theo qui luật sau:

+ Mỗi dòng đều bắt đầu và kết thúc bởi số 1.

+ Phần tử thứ j ở dòng k nhận được bằng cách cộng 2 phần tử thứ j-1 và j ở dòng thứ k-1.

```
Uses Crt;
Var Dong:Array[0..20] Of Byte;
   n,i,j:Byte;
Begin
  Write('n= '); Readln(n);
  Clrscr;
  Dong[0]:=1;
  Writeln(Dong[0]:4);

  {Khoi tao gia tri cua dong}
  For i:=1 To n Do Dong[i]:=0;
 {Voi moi dong i}
  For i:=1 To n Do
   Begin
     For j:=i DownTo 1 Do
      Begin
        Dong[j]:=Dong[j-1]+Dong[j];
        Write(Dong[j]:4);
      End;
     Writeln(Dong[i]:4);
   End;
  Readln;
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 5.13: Nhập vào một mảng các số nguyên.

a/ Xếp lại mảng đó theo thứ tự giảm dần.

b/ Nhập vào một số nguyên từ bàn phím. Chèn số đó vào mảng sao cho mảng vẫn có thứ tự giảm dần. (không được xếp lại mảng)

Gợi ý:

- Tìm vị trí cần chèn: i.
- Đẩy các phần tử từ vị trí i tới n sang phải 1 vị trí.
- Gán: A[i]=x;

Bài tập 5.14: Cho 2 mảng số nguyên: Mảng A có m phần tử, mảng B có n phần tử.

a/ Sắp xếp lại các mảng đó theo thứ tự giảm dần.

b/ Trộn 2 mảng đó lại thành mảng C sao cho mảng C vẫn có thứ tự giảm dần (Không được xếp lại mảng C).

Gợi ý:

- Dùng 2 chỉ số i,j để duyệt qua các phần tử của 2 mảng A, B và k là chỉ số cho mảng C.
- Trong khi (i<=m) và (j<=n) thì:

  {Tức là khi đồng thời cả 2 dãy A, B đều chưa duyệt hết}

  + Nếu A[i]>B[j] thì: C[k]:=A[i]; i:=i+1;
  + Ngược lại: C[k]:=B[j]; j:=j+1;
- Nếu dãy nào hết trước thì đem phần còn lại của dãy kia bổ sung vào cuối dãy C.

Bài tập 5.15: Viết chương trình tính tổng và tích 2 ma trận vuông A, B cấp n.

Gợi ý:

Công thức tính tổng 2 ma trận: $\mathbf{C_{ij} = A_{ij} + B_{ij}}$

Công thức tính tích 2 ma trận: $\mathbf{C_{ij}} = \sum_{k} * B_{kj}$

Bài tập 5.16: Viết chương trình nhập vào 2 dãy số nguyên $(a)_n$ và $(b)_m$, m[redacted]. Kiểm tra xem dãy {b} có phải là dãy con của dãy {a} không?

Bài tập 5.17: Viết chương trình nhập vào một dãy số nguyên $a_1, a_2, ..., a_n$. Tìm trong dãy {a} một dãy con tăng dần dài nhất (có số phần tử lớn nhất) và in ra màn hình dãy con đó.

Bài tập 5.18: Cho mảng 2 chiều A cấp mxn. Viết chương trình sắp xếp lại mảng A theo yêu cầu sau:

a/ Các phần tử trên mỗi dòng được sắp xếp theo thứ tự giảm dần.

b/ Các dòng được sắp xếp lại theo thứ tự tăng dần của tổng các phần tử trên mỗi dòng.

Chương 6
XÂU KÝ TỰ (STRING)

I. KHAI BÁO KIỂU STRING

TYPE TênKiểu = STRING[Max];

VAR Tên biến : TênKiểu;

hoặc khai báo biến trực tiếp:

VAR Tên biến : STRING[Max];

Trong đó Max là số ký tự tối đa có thể chứa trong chuỗi (Max [redacted]0,255]). Nếu không có khai báo [Max] thì số ký tự mặ mặc định trong chuỗi là 255.

Ví dụ:

```
Type    Hoten  = String[30];
        St80 = String[80];
Var     Name : Hoten;
        Line : St80;
        St : String; {St có tối đa là 255 ký tự}
```

II. TRUY XUẤT DỮ LIỆU KIỂU STRING

- Có thể sử dụng các thủ tục xuất nhập Write, Writeln, Readln để truy xuất các biến kiểu String.
- Để truy xuất đến ký tự thứ k của xâu ký tự, ta sử dụng cú pháp sau: **Tênbiến[k].**

III. CÁC PHÉP TOÁN TRÊN XÂU KÝ TỰ

3.1. Phép nối xâu: +

3.2. Các phép toán quan hệ: =, <>, <, <=, >, >=.

Chú ý: *Các phép toán quan hệ được so sánh theo thứ tự từ điển.*

IV. CÁC THỦ TỤC VÀ HÀM VỀ XÂU KÝ TỰ

4.1. Hàm lấy chiều dài của xây ký tự

LENGTH(St : String):Integer;

4.2. Hàm COPY(St : String; Pos, Num: Byte): String;

Lấy ra một xâu con từ trong xâu St có độ dài Num ký tự bắt đầu từ vị trí Pos .

4.3. Hàm POS(SubSt, St :String):Byte;

Kiểm tra xâu con SubSt có nằm trong xâu St hay không? Nếu xâu SubSt nằm trong xâu St thì hàm trả về vị trí đầu tiên của xâu con SubSt trong xâu St, ngược lại hàm trả về giá trị 0.

4.4. Thủ tục DELETE(Var St:String; Pos, Num: Byte);

Xoá trong xâu St Num ký tự bắt đầu từ vị trí Pos.

4.5. Thủ tục INSERT(SubSt: String; Var St: String; Pos: Byte);

Chèn xâu SubSt vào xâu St bắt đầu tại vị trí Pos.

4.6. Thủ tục STR(Num; Var St:String);

Đổi số nguyên hay thực Num thành dạng xâu ký tự, kết quả lưu vào biến St.

4.7. Thủ tục VAL(St:String; Var Num; Var Code:Integer);

Đổi xâu số St thành số và gán kết quả lưu vào biến Num. Nếu việc chuyển đổi thành công thì biến Code có giá trị là 0, ngược lại biến Code có giá trị khác 0 (vị trí của lỗi).

BÀI TẬP MẪU

Bài tập 6.1: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Đổi xâu ký tự đó sang chữ in hoa rồi in kết quả ra màn hình.

Ví dụ :Xâu abcdAbcD sẽ cho ra xâu ABCDABCD.

```
Uses Crt;
Var  St:String;
     i:Byte;
```

```
Begin
   Write('Nhap xau St: '); Readln(St);
   For i:=1 to length(St) do St[i]:=Upcase(St[i]);
   Write('Xau ket qua: ', St);
   Readln;
End.
```

Bài tập 6.2: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Đổi xâu ký tự đó sang chữ thường rồi in kết quả ra màn hình.

Ví dụ :Xâu abCdAbcD sẽ cho ra xâu abcdabcd.

```
Uses Crt;
Var   St:String;
      i:Byte;
Begin
   Write('Nhap xau St: '); Readln(St);
   For i:=1 to length(St) do
      If St[i] IN ['A'..'Z'] Then St[i]:=CHR(ORD(St[i])+32);
   Write('Xau ket qua: ', St);
   Readln;
End.
```

Bài tập 6.3: Viết chương trình đếm số ký tự chữ số trong một xâu ký tự được nhập vào từ bàn phím.

```
Uses Crt;
Var   St:String;
      i,d:Byte;
Begin
   Write('Nhap xau St: '); Readln(St);
   For i:=1 to length(St) do
      If St[i] IN ['0'..'9'] Then d:=d+1;
   Write('So ky tu chu so trong xau: ', d);
   Readln;
End.
```

Bài tập 6.4: Viết chương trình nhập một xâu từ bàn phím. In ra xâu đó sau khi xóa hết các ký tự trắng thừa trong xâu. (Ký tự trắng thừa là các ký tự trắng đầu xâu, cuối xâu và nếu ở giữa xâu có 2 ký tự trắng liên tiếp nhau thì có 1 ký tự trắng thừa).

```
Uses Crt;
Var   St:String;
Procedure  XoaTrangThua(Var St:String);
Begin
   {Xóa các ký tự trắng ở đầu xâu}
   While St[1]=#32 Do Delete(St,1,1);
   {Xóa các ký tự trắng ở cuối xâu}
   While St[Length(St)]=#32 Do Delete(St,Length(St),1);
   {Xóa các ký tự trắng ở giữa xâu}
   While POS(#32#32,St)<>0 Do Delete(St,POS(#32#32,St),1);
End;
Begin
   Write('Nhap xau St: '); Readln(St);
```

```
   XoaTrangThua(St);
   Write('Xau sau khi xoa cac ky tu trang thua: ', St);
   Readln;
End.
```

Bài tập 6.5: Viết chương trình liệt kê các từ của một xâu ký tự được nhập vào từ bàn phím, mỗi từ phải được viết trên một dòng.

```
Uses Crt;
Var   St:String;
Procedure  XoaTrangThua(Var St:String);
Begin
   {Xóa các ký tự trắng ở đầu xâu}
   While St[1]=#32 Do Delete(St,1,1);
   {Xóa các ký tự trắng ở cuối xâu}
   While St[Length(St)]=#32 Do Delete(St,Length(St),1);
   {Xóa các ký tự trắng ở giữa xâu}
   While POS(#32#32,St)<>0 Do Delete(St,POS(#32#32,St),1);
End;
Begin
   Write('Nhap xau St: '); Readln(St);
   XoaTrangThua(St);
   St:=St+#32;
   Writeln('Liet ke cac tu trong xau: ');
   While POS(#32,St)<>0 Do
      Begin
         Writeln(Copy(St,1,POS(#32,St)));
         Delete(St,1,POS(#32,St));
      End;
   Readln;
End.
```

Bài tập 6.6: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Tìm xâu đảo ngược của xâu đó rồi in kết quả ra màn hình theo 2 cách: Đệ qui và không đệ qui.

Ý tưởng:

- Nếu xâu St có 1 ký tự thì xâu đảo = St.
- Ngược lại: Xâu đảo = Ký tự cuối + Đệ qui(Phần còn lại của xâu St).

```
Uses Crt;
Var   St:String;
{Giải thuật không đệ qui}
Function XauDao(St:String):String;
Var   S:String;
      i:Byte;
Begin
   S:='';
   For i:=Length(St) DowTo 1 Do S:=S+St[i];
   XauDao:=S;
End;
```

```
{Giải thuật đệ qui}
Function DeQui(St:String):String;
Begin
   If Length(St)<=1 Then DeQui:=St
   Else DeQui:=St[Length(St)] + DeQui(Copy(St,1,Length(St)-1));
End;
Begin
   Write('Nhap xau St: '); Readln(St);
   Write('Xau dao nguoc: ', XauDao(St));
   Readln;
End.
```

Bài tập 6.7: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Thông báo lên màn hình các chữ cái có trong xâu và số lượng của chúng (Không phân biệt chữ hoa hay chữ thường).

Ý tưởng:

- Dùng một mảng dem với chỉ số là các chữ cái để lưu trữ số lượng của các chữ cái trong xâu.
- Duyệt qua tất cả các ký tự của xâu St: Nếu ký tự đó là chữ cái thì tăng ô biến mảng dem[St[i]] lên 1 đơn vị.

```
Uses Crt;
Var   St:String;
      dem: Array['A'..'Z'] Of Byte;
      i:Byte;
      ch:Char;
Begin
   Write('Nhap xau St: '); Readln(St);
   {Khởi tạo mảng}
   For ch:='A' To 'Z' Do dem[ch]:=0;
   {Duyệt xâu}
   For i:=1 To Length(St) Do
      If Upcase(St[i]) IN ['A'..'Z'] Then Inc(dem[Upcase(St[i])]);
   {Liệt kê các ký tự ra màn hình}
   For ch:='A' To 'Z' Do
      If dem[ch]>0 Then Writeln(ch,' : ',dem[ch]);
   Readln;
End.
```

Bài tập 6.8: Viết chương trình xóa các ký tự chữ số trong một xâu ký tự được nhập vào từ bàn phím.

```
Uses Crt;
Var   St:String;
{Hàm POSNUM kiểm tra xem trong xâu St có ký tự chữ số hay không? Nếu có, hàm trả về vị trí đầu tiên của ký tự chữ số, ngược lại hàm trả về giá trị 0}
Function POSNUM(St:String):Byte;
Var   OK:Boolean;
      i:Byte;
Begin
   OK:=False;
   i:=1;
```

```
    While (i<=Length(St)) AND (Not OK) Do
       If St[i] IN ['0'..'9'] Then OK:=True
       Else i:=i+1;
    If OK Then POSNUM:=i Else POSNUM:=0;
End;
Begin
    Write('Nhap xau St: '); Readln(St);
    While POSNUM(St)<>0 Do Delete(St,POSNUM(St),1);
    Write('Xau sau khi xoa: ',St);
    Readln;
End.
```

Bài tập 6.9: Viết chương trình để mã hoá và giải mã một xâu ký tự bằng cách đảo ngược các bit của từng ký tự trong xâu.

```
Uses crt;
Var st:string;
{Hàm đảo bit ký tự c}
Function DaoBit(c:char):char;
Var n,i,s,bitcuoi,Mask:byte;
Begin
 {Đổi ký tự sang số}
  n:=ORD(c);
 {s: kết quả đảo bit, Mask: mặt nạ dùng để bật bit thứ i}
  s:=0;
  Mask:=128;
For i:=1 To 8 Do {duyệt qua 8 bit của n}
   Begin
     {Lấy bit cuối cùng của n: bit cực phải}
     bitcuoi:=n AND 1;
     n:=n shr 1; {loại bỏ bit cuối cùng: n:=n DIV 2}
     {Bật bit thứ i lên: từ trái sang phải}
     if bitcuoi=1 then s:=s OR Mask;
     Mask:=Mask shr 1; { Mask:= Mask DIV 2}
   End;
    DaoBit:=CHR(s);
End;

Function MaHoa(st:string):string;
Var i:Byte;
Begin
 {Đảo bit từng ký tự trong xâu st}
  For i:=1 To Length(st) Do st[i]:=DaoBit(st[i]);
  Mahoa:=st;
End;
Begin
 Write('Nhap xau: '); Readln(st);
 st:=MaHoa(st);
 Writeln('Xau sau khi ma hoa: ',st);
 Readln;
 st:=MaHoa(st);
```

```
 Writeln('Xau sau khi giai ma: ',st);
 Readln;
End.
```

Bài tập 6.10: Viết chương trình thực hiện phép cộng 2 số tự nhiên lớn (không quá 255 chữ số).

```
Uses crt;
Var so1,so2,kqua:string;
Procedure LamDayXau(Var st1,st2:string);
{Them so 0 vao truoc xau ngan}
var i:Byte;
Begin
  If Length(st1)>Length(st2) Then
    For i:=1 To Length(st1)-Length(st2) Do st2:='0'+st2
  Else
    For i:=1 To Length(st2)-Length(st1) Do st1:='0'+st1;
End;
Function Cong(st1,st2:string):string;
Var i,a,b,c,sodu:Byte;
    code:integer;
    st,ch:string;
Begin
  st:=''; sodu:=0;
  LamDayXau(st1,st2);
  {Lấy từng số của 2 xâu: từ phải sang trái}
  For i:=Length(st1) DownTo 1 Do
   Begin
     {Đổi ký tự sang số nguyên}
     Val(st1[i],a,code);
     Val(st2[i],b,code);
     {Tính tổng của 2 số a,b vừa lấy ra cho vào biến c}
     c:=(a+b+sodu) MOD 10;
     {Lấy phần dư của tổng a+b}
     sodu:=(a+b+sodu) DIV 10;
     {Đổi số nguyên c sang xâu ký tự ch}
     str(c,ch);
     {Cộng xâu ch vào bên trái xâu kết quả st}
     st:=ch+st;
   End;
{Xử lý trường hợp số dư cuối cùng >0}
   If sodu>0 Then
    Begin
     str(sodu,ch);
     st:=ch+st;
    End;
   Cong:=st;
End;
Begin
  Write('Nhap so thu nhat: '); Readln(so1);
  Write('Nhap so thu hai: '); Readln(so2);
  kqua:=Cong(so1,so2);
  Writeln('Tong= ',kqua);
```

```
  Readln;
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 6.11: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Tìm và in ra màn hình một từ có độ dài lớn nhất trong xâu.

Gợi ý:

Tách từng từ để so sánh (xem bài tập 5).

Bài tập 6.12: Viết chương trình nhập một xâu ký tự St từ bàn phím và một ký tự ch. In ra màn hình xâu St sau khi xóa hết các ký tự ch trong xâu đó.

Gợi ý:

```
While POS(ch,st)<>0 Do Delete(st,POS(ch,st),1);
```

Bài tập 6.13: Viết chương trình nhập một xâu vào từ bàn phím và thông báo lên màn hình xâu đó có phải đối xứng không theo 2 cách: Đệ qui và không đệ qui. (Ví dụ: abba, abcba là các xâu đối xứng).

Gợi ý:

- Nếu xâu Length(st)<=1 thì st là xâu đối xứng
- Ngược lại:
  + Nếu st[1]<>st[Length(st)] thì st không đối xứng
  + Ngược lại: Gọi đệ qui với xâu st sau khi bỏ đi ký tự đầu và ký tự cuối.

Bài tập 6.14: Viết chương trình đảo ngược thứ tự các từ trong một xâu được nhập vào từ bàn phím.

Ví dụ: Xâu ***Nguyen Van An*** sẽ thành ***An Van Nguyen.***

Gợi ý:

Tách từng từ nối vào đầu xâu mới (xem bài tập 5).

Bài tập 6.15: Viết chương trình nhập vào 2 xâu ký tự s1 và s2. Kiểm tra xem xâu s2 xuất hiện bao nhiêu lần trong xâu s1. (Lưu ý: length(s2)<= length(s1)).

Gợi ý:

Dùng hàm POS để kiểm tra và thủ tục DELETE để xóa bớt sau mỗi lần kiểm tra.

Bài tập 6.16: Viết chương trình nhập vào một dòng văn bản, hiệu chỉnh văn bản theo những yêu cầu sau đây và in văn bản sau khi hiệu chỉnh ra màn hình:

a. Xóa tất cả các ký tự trắng thừa.

b. Trước các dấu câu không có các ký tự trắng, sau các dấu câu có một ký tự trắng.

c. Đầu câu in hoa.

Bài tập 6.17: Viết chương trình thực hiện phép nhân 2 số nguyên lớn.

Gợi ý:

- Viết hàm để nhân một số lớn với số có 1 chữ số.
- Áp dụng hàm tính tổng 2 số lớn (xem bài tập 10).

Bài tập 6.18: Viết chương trình để nén và giải nén một xâu ký tự .

Ví dụ: Xâu ‘AAAABBBCDDDDDDDEEF’ sau khi nén sẽ trở thành ‘4A3BC7D2EF’.

Bài tập 6.19: Viết chương trình nhập vào họ tên đầy đủ của các học viên một lớp học (không quá 50 người). Hãy sắp xếp lại họ tên của các học viên đó theo thứ tự Alphabet (Nếu tên trùng nhau thì xếp thứ tự theo họ lót, nếu họ lót cũng trùng nhau thì xếp thứ tự theo họ). In ra màn hình danh sách của lớp học sau khi đa sắp xếp theo thứ tự Alphabet.

Gợi ý:

- Dùng mảng xâu ký tự để lưu trữ họ tên học viên.
- Đảo ngược các từ của họ tên trước khi sắp xếp.

Chương 7
KIỂU BẢN GHI (RECORD)

I. KHAI BÁO DỮ LIỆU KIỂU RECORD

```
TYPE   TênKiểu = RECORD
          Field1 : Kiểu1;
          Field2 : Kiểu2;
          ...
          FieldN: KiểuN;
       END;
VAR    Biến : TênKiểu;
```

Ví dụ:

```
TYPE   HocSinh  = Record
          Hoten : String[20];
          Tuoi : Integer;
          DiemTB : real;
       End;
VAR  HS : HocSinh;
```

II. XUẤT NHẬP DỮ LIỆU KIỂU RECORD

Không thể dùng các thủ tục xuất/nhập, các phép toán so sánh đối với các biến kiểu record mà chỉ có thể sử dụng thông qua từng trường của biến record đó.

2.1. Truy nhập trực tiếp: TênbiếnRecord.Field

2.2. Sử dụng câu lệnh WITH

```
WITH  TênbiếnRecord DO
   BEGIN
      Xử lý Field1;
      Xử lý Field2;
      ...
      Xử lý FieldN;
   END;
```

2.3. Gán biến Record: Ta có thể gán 2 biến Record cùng kiểu với nhau.

BÀI TẬP MẪU

Bài tập 7.1: Viết chương trình thực hiện phép cộng 2 số phức.

```
Uses Crt;
Type  Complex =  Record
                    a,b:Real;
                 End;
Var   c1,c2,c3:Complex;
      dau:string;
Begin
   Writeln('Nhap so phuc c1:');
   Write('Phan thuc a = '); Readln(c1.a);
```

```
    Write('Phan ao b = '); Readln(c1.b);
     Writeln('Nhap so phuc c2:');
    Write('Phan thuc a = '); Readln(c2.a);
    Write('Phan ao b = '); Readln(c2.b);
    {Tính tổng 2 số phức}
    c3.a := c1.a + c2.a;
    c3.b := c1.b + c2.b;
    {In kết quả ra màn hình}
    Writeln('Tong cua 2 so phuc:');
    If c1.b>=0 Then dau:='+i' else dau:='-i';
    Writeln('c1 = ', c1.a:0:2, dau, abs(c1.b):0:2); {Số phức c1}
    If c2.b>=0 Then dau:='+i' else dau:='-i';
    Writeln('c2 = ', c2.a:0:2, dau, abs(c2.b):0:2); {Số phức c2}
    Writeln('La so phuc:');
    If c3.b>=0 Then dau:='+i' else dau:='-i';
    Writeln('c3 = ', c3.a:0:2, dau, abs(c3.b):0:2); {Số phức c3}
    Readln;
End.
```

Bài tập 7.2: Viết chương trình quản lý điểm thi Tốt nghiệp của sinh viên với 2 môn thi: Cơ sở và chuyên ngành. Nội dung công việc quản lý bao gồm:

- Nhập điểm cho từng sinh viên.
- In danh sách sinh viên ra màn hình.
- Thống kê số lượng sinh viên thi đậu.
- In ra màn hình hình danh sách những sinh viên bị thi lại.

```
Uses Crt;
Const Max=200;
Type SinhVien=Record
              Hoten:string[30];
              DiemCS,DiemCN:Byte;
        End;
Var SV:ARRAY[1..Max] Of SinhVien;
     n:Byte;
     c:Char;
Procedure NhapDanhSach;
Var ch:Char;
Begin
  Clrscr;
  Writeln('NHAP DANH SACH SINH VIEN');
  n:=0;
  Repeat
    n:=n+1;
    With SV[n] Do
     Begin
       Write('Ho ten: '); Readln(Hoten);
```

```
      Write('Diem co so: '); Readln(DiemCS);
      Write('Diem chuyen nganh: '); Readln(DiemCN);
     End;
    Writeln('Nhan phim bat ky de nhap tiep/Nhan <ESC> de ket thuc!');
    ch:=Readkey;
  Until ch=#27;
End;
Procedure InDanhSach;
Var ch:Char;
     i:Byte;
Begin
  Clrscr;
  Writeln('DIEM THI TOT NGHIEP SINH VIEN');
  Writeln;
  WRITELN('STT      Ho ten       Diem Co so    Diem Chuyen nganh');
  For i:=1 To n do
   With SV[i] Do
    Begin
     Writeln(i:3,'.',Hoten:20,DiemCS:5,DiemCN:20);
    End;
  ch:=ReadKey;
End;
Procedure DanhSachSVThilai;
Var ch:Char;
     i:Byte;
Begin
  Clrscr;
  Writeln('DANH SACH SINH VIEN THI LAI');
  Writeln;
  WRITELN('STT      Ho ten       Diem Co so    Diem Chuyen nganh');
  For i:=1 To n do
   With SV[i] Do
    Begin
     If (DiemCS<5)OR(DiemCN<5) Then
      Writeln(i:3,'.',Hoten:20,DiemCS:5,DiemCN:20);
    End;
  ch:=ReadKey;
End;
Procedure ThongKeSVThiDau;
Var S,i:Byte;
     ch:Char;
Begin
  S:=0;
  For i:=1 To n Do
   If (SV[i].DiemCS>=5)AND(SV[i].DiemCN>=5) Then S:=S+1;
  Writeln('So sinh vien thi dau la: ',s);
  ch:=Readkey;
End;
Begin
  Repeat
```

```
    Clrscr;
    Writeln('CHUONG TRINH QUAN LY DIEM THI TOT NGHIEP SINH VIEN');
    Writeln('1. Nhap danh sach sinh vien');
    Writeln('2. In danh sach sinh vien');
    Writeln('3. Thong ke so sinh vien thi dau');
    Writeln('4. danh sach sinh vien thi lai');
    Writeln('<ESC>: Thoat');
    c:=Readkey;
    Case c Of
    '1': NhapDanhSach;
    '2': InDanhSach;
    '3': ThongKeSVThiDau;
    '4': DanhSachSVThilai;
    End;
  Until c=#27;
End.
```

Bài tập 7.3: Viết chương trình nhập vào n đỉnh của một đa giác lồi S.

a/ Tính diện tích của S biết:

$$dt(S)=\frac{1}{2}\left|\sum_{i}(x_i y_{i+1} - x_{i+1} y_i)\right|$$

trong đó: (x_i,y_i) là tọa độ đỉnh thứ i của đa giác S.

b/ Nhập vào thêm một điểm P(x,y). Hãy kiểm tra xem P nằm trong hay ngoài đa giác S.

Ý tưởng:

Nối P với các đỉnh của đa giác S thì ta được n tam giác: $S_i = PP_iP_{i+1}$, với $P_{n+1}=P_1$.

Nếu $\sum_{i}(S_i) = dt(S)$ thì P .

```
Uses Crt;
Type Toado=Record
           x,y:integer;
         end;
     Mang=array[0..30] of Toado;
Var n:Byte;
    A:Mang;
    P:ToaDo;
Procedure NhapDinh(var n:Byte; Var P:Mang);
Var i:Byte;
Begin
  Write('Nhap so dinh cua da giac n = '); readln(n);
  For i:=1 to n do
    Begin
      Write('P[',i,'].x = ');readln(P[i].x);
      Write('P[',i,'].y = ');readln(P[i].y);
    End;
End;
Function DienTichDaGiac(n:Byte;P:Mang):real;
Var i,j:integer;
```

```
    s:real;
Begin
  s:=0;
  for i:= 1 to n do
    begin
      if i=n then j:=1 else j:=i+1;
      s:=s+((P[i].x*P[j].y-P[j].x*P[i].y));
    end;
  DienTichDaGiac:=abs(s)/2;
end;
Function DienTichTamGiac(A,B,C:ToaDo):real;
Begin
  DienTichTamGiac:=abs(A.x*B.y-B.x*A.y+B.x*C.y-C.x*B.y+C.x*A.y-A.x*C.y)/2;
End;
Function KiemTra(PP:ToaDo;n:Byte;P:Mang):Boolean;
Var i,j:integer;
    s:real;
begin
  s:=0;
  For i:=1 to n do
    begin
      if i=n then j:=1 else j:=i+1;
      s:=s+DienTichTamGiac(PP,P[i],P[j]);
    end;
  If round(s)=round(DienTichDaGiac(n,P)) then KiemTra:=true
  else KiemTra:=false;
end;
Begin
  NhapDinh(n,A);
  Writeln('S=',DienTichDaGiac(n,A):0:2);
  Readln;
  Writeln('Nhap diem P:');
  Write('P.x = ');readln(P.x);
  Write('P.y = ');readln(P.y);
  If KiemTra(P,n,A) Then Writeln('Diem P nam trong da giac S.')
  Else Writeln('Diem P nam ngoai da giac S.');
  Readln;
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 7.4: Viết chương trình nhân hai số phức c1, c2.

Bài tập 7.5: Viết chương trình quản lý điểm thi học phần của sinh viên bao gồm các trường sau: Họ tên, Điểm Tin, Điểm ngoại ngữ, Điểm trung bình, Xếp loại. Thực hiện các công việc sau:

a/ Nhập vào danh sách sinh viên của một lớp (không quá 30 người), bao gồm: Họ tên, Điểm Tin, Điểm Ngoại ngữ. Tính Điểm trung bình và Xếp loại cho từng sinh viên.

b/ In ra màn hình danh sách sinh viên của lớp đó theo dạng sau:

Họ tên	Điểm Tin	Điểm Ngoại ngữ	Điểm T.Bình	Xếp loại
Trần Văn An	8	9	8.5	Giỏi
Lê Thị Béo	7	5	6.0	T.Bình
.....	.....	.....	.....	.....

c/ In ra màn hình danh sách những sinh viên phải thi lại (nợ một trong hai môn).

d/ In ra danh sách những sinh viên xếp loại Giỏi.

e/ Tìm và in ra màn hình những sinh viên có điểm trung bình cao nhất lớp.

f/ Sắp xếp lại danh sách sinh viên theo thứ tự Alphabet.

g/ Sắp xếp lại danh sách sinh viên theo thứ tự giảm dần của điểm trung bình.

h/ Viết chức năng tra cứu theo tên không đầy đủ của sinh viên. Ví dụ: Khi nhập vào tên **Phuong** thì chương trình sẽ tìm và in ra màn hình thông tin đầy đủ của những sinh viên có tên **Phuong** (chẳng hạn như: **Pham Anh Phuong, Do Ngoc Phuong, Nguyen Nam Phuong...**).

Bài tập 7.6: Viết chương trình quản lý sách ở thư viện gồm các trường sau: Mã số sách, Nhan đề, Tên Tác giả, Nhà Xuất bản, Năm xuất bản.

a/ Nhập vào kho sách của thư viện (gồm tất cả các trường).

b/ In ra màn hình tất cả các cuốn sách có trong thư viện.

c/ Tìm một cuốn sách có mã số được nhập vào từ bàn phím. Nếu tìm thấy thì in ra màn hình thông tin đầy đủ của cuốn sách đó, ngược lại thì thông báo không tìm thấy.

c/ Tìm và in ra màn hình tất cả các cuốn sách có cùng tác giả được nhập vào từ bàn phím.

d/ Lọc ra các cuốn sách được xuất bản trong cùng một năm nào đó.

e/ Tìm và in ra màn hình các cuốn sách mà nhan đề có chứa từ bất kỳ được nhập vào từ bàn phím.

Chương 8
DỮ LIỆU KIỂU FILE

I. KHAI BÁO

Type <Tên kiểu File> = **File of** <Kiểu phần tử>;
Var <Tên biến File> : <Tên kiểu File>;

hoặc khai báo trực tiếp:

Var <Tên biến File> : **File of** <Kiểu phần tử>;

Ví dụ:

```
Type   SanPham = File of Record
                    Ten: String[20];
                    SoHieu: Byte;
                 End;
Var f,g: SanPham;
```

hoặc khai báo trực tiếp:

```
Var f,g: File of  Record
                    Ten: String[20];
                    SoHieu: Byte;
                 End;
```

Chú ý:

- Pascal theo dõi các thao tác truy nhập thông qua con trỏ file. Mỗi khi một phần tử nào đó được ghi vào hay đọc từ file, con trỏ của file này được tự động chuyển đến phần tử tiếp theo.
- Các biến kiểu file không được phép có mặt trong phép gán hoặc trong các biểu thức.

II. CÁC THỦ TỤC VÀ HÀM CHUẨN

2.1. Các thủ tục chuẩn

2.1.1. Gán tên file

Cú pháp: **Assign(F, Filename);**

Chức năng: Gán một file trên đĩa có tên là Filename cho biến file F, mọi truy xuất trên file cụ thể được thực hiện thông qua biến file này.

Chú ý:

Filename bao gồm cả tên ổ đĩa và đường dẫn nếu file không nằm trong ổ đĩa, thư mục hiện thời.

2.1.2. Mở file mới

Cú pháp: **Rewrite(F);**

Chức năng: Tạo file mới có tên đã gán cho biến file F. Nếu file đã có trên đĩa thì mọi dữ liệu trên đó sẽ bị xoá và con trỏ file trỏ ở vị trí đầu tiên của file.

2.1.3. Mở file đã có trên đĩa

Cú pháp: **Reset(F);**

Chức năng: Mở file có tên đã gán cho biến file F. Nếu file chưa có trên đĩa thì chương trình sẽ dừng vì gặp lỗi xuất/nhập.

***Chú ý:* Kiểm tra khi mở file**

{$I+}: Mở việc kiểm tra. Khi gặp lỗi Vào/ra chương trình sẽ báo lỗi và dừng lại

{$I-}: Không kiểm tra Vào/ra, chương trình không dừng lại nhưng treo các thủ tục Vào/ra khác cho đến khi hàm **IOresult** (hàm chuẩn của PASCAL). Hàm trả về giá trị **true** nếu việc mở file xảy ra tốt đẹp.

Ví dụ:

```
Procedure MoFile;
Var  ok:Boolean;
     St:String;
     F:Text;
Begin
```

```
   Repeat
      Write('Nhập tên tệp: ');readln(st);
      Assign(F,st);
      {$I-} (*Chuyển việc kiểm tra vào ra cho người dùng*)
      Reset(F);
      Ok:=IOResult;
      {$I+}
      if not OK then writeln('Không mở được ');
   Until OK;
End;
```

2.1.4. Đọc dữ liệu từ file

Cú pháp: **Read(F, x);**

Chức năng: Đọc một phần tử dữ liệu từ file F ở vị trí con trỏ file và gán cho các biến x.

2.1.5. Ghi dữ liệu lên file

Cú pháp: **Write(F, Value);**

Chức năng: Ghi giá trị Value vào file F tại vị trí hiện thời của con trỏ file.

2.1.6. Di chuyển con trỏ file

Cú pháp: **Seek(F, n);**

Chức năng: Di chuyển con trỏ file đến phần tử thứ n (phần tử đầu tiên có thứ tự là 0).

2.1.7. Đóng file

Cú pháp: **Close(F);**

Chức năng: Cập nhật mọi sửa đổi trên file F và kết thúc mọi thao tác trên file này.

2.1.8. Xoá file

Cú pháp: **Erase(F);**

Chức năng: Xoá file trên đĩa có tên gán đã được gán cho biến file F (file cần xoá là file đang đóng).

2.1.9. Đổi tên file

Cú pháp: **Rename(F, NewFile);**

Chức năng: Đổi tên của file đang gán cho biến file F thành tên file mới là NewFile.

2.2. Các hàm chuẩn

2.2.1. Hàm trả về vị trí con trỏ file

Cú pháp: **Filepos(F);**

Chú ý: Con trỏ ở đầu file tương ứng vị trí 0.

2.2.2. Hàm kiểm tra cuối file

Cú pháp: **EOF(F);**

Chức năng: Hàm trả về giá trị ***True*** nếu con trỏ file đang ở cuối file, ngược lại hàm trả về giá trị ***False.***

2.2.3. Hàm trả về kích thước của file

Cú pháp: **FileSize(F);**

Chức năng: Hàm trả về ***số lượng phần tử*** có trong file.

III. FILE VĂN BẢN (TEXT FILE)

Thành phần cơ bản là ký tự, song có thể được cấu trúc thành các dòng, mỗi dòng được kết thúc bởi CR và LF, CR có mã ASCII là 13 và LF có mã 10. Cuối file sẽ có dấu kết thúc file Ctrl-Z có mã là 26.

Do các dòng có độ dài thay đổi nên không tính trước được vị trí của một dòng trong file. Vì vậy file dạng Text chỉ có thể đệoc xử lý một cách tuần tự.

3.1. Khai báo

Var **<Tên biến file>: Text;**

3.2. Các thủ tục và hàm chỉ tác động trên file dạng text

3.2.1. Thủ tục Append

Cú pháp: **Append(F);**

Chức năng: Mở file đã tồn tại để bổ sung nội dung vào cuối file.

3.2.2. Thủ tục Readln

Cú pháp: **Readln(F,x);**

Chức năng: Đọc một dòng từ vị trí con trỏ file và gán cho biến **x**. Thực hiện xong, con trỏ file sẽ chuyển về đầu dòng tiếp theo. Biến **x** có thể nhận các kiểu: Char, String hoặc kiểu số.

3.2.3. Thủ tục Writeln

Cú pháp: **Writeln(F, x);**

Chức năng: Ghi giá trị x vào file ở vị trí con trỏ file. Kết thúc thủ tục, con trỏ file sẽ chuyển về đầu dòng sau.

Chú ý:

Máy in được xem là một file dạng text, và biến được mở sẵn trong Unit Printer cho file này là LST. Vì vậy để in một dòng St ra máy in ta có thể dùng lệnh Writeln(LST,St).

3.2.4. Thủ tục Flush

Cú pháp: **Flush(F);**

Chức năng: Cập nhật nội dung của file có tên gán cho biến file F mà không cần dùng thủ tục Close và vẫn có thể thao tác trên file.

3.2.5. Thủ tục SetTextBuf

Cú pháp: **SetTextBuf(F, x);**

Chức năng: Thay đổi vùng nhớ đệm dành cho file dạng text với kích thước cho bởi biến x. Mặc định vùng nhớ này là 128 byte.

Chú ý:

Thủ tục này phải được gọi trước các thủ tục mở file: Reset, Rewrite, Append.

3.2.6. Hàm EOLn

Cú pháp: **EOLn(F);**

Chức năng: Hàm trả về giá trị ***True*** nếu con trỏ đang ở cuối một dòng, ngược lại hàm trả về giá trị ***False***.

***Chú ý*:**

Các thủ tục và hàm không sử dụng được đối với file dạng text: ***Seek, FilePos, FileSize***.

Sau đây là các thao tác cơ bản khi xuất nhập file:

Ghi dữ liệu vào file	Đọc dữ liệu từ file

ASSIGN(f,FileName);
REWRITE(f);
...
WRITE(f,value);
...
CLOSE(f);
	ASSIGN(f,FileName);
	RESET(f);
	...
	While Not EOF(f) Do
	Begin
	READ(f,x);
	...
	End;
	...
	CLOSE(f);

IV. FILE KHÔNG ĐỊNH KIỂU (FILE VẬT LÝ)

4.1. Khái niệm

File không định kiểu là file không xác định kiểu của mỗi thành phần trong file, mà được hiểu là một dãy byte, mỗi phần tử có kích thước k byte, quy định bởi người lập trình. File không định kiểu tương hợp với mọi kiểu file.

4.2. Khai báo

Var **<Tên biến File>: File;**

4.3. Các thủ tục và hàm có thể thao tác trên file không đinh kiểu

4.3.1. Mở file

Mở file chưa có trên đĩa: **Rewrite(F, k);**

Mở file đã có trên đĩa: **Reset(F, k);**

Giá trị k mô tả số lượng byte sẽ được đọc ghi trong một thao tác. Kích thước của file phải là bội số của k.

4.3.2. Xuất/ nhập dữ liệu

Cú pháp: **BlockRead(F, x, n [,Kq]);**
BlockWrite(F, x, n [,Kq]);

Chức năng:

- Đọc/ Ghi n "bản ghi". Mỗi "bản ghi" được hiểu là một phần tử k byte.
- x chứa nội dung đọc/ghi
- Kq là số lương"bản ghi" được thực hiện.

Chú ý:

File không định kiểu thường được dùng trong các thao tác sao chép với tốc độ cao.

BÁI TẬP MẪU

Bài tập 8.1: Tạo một file SINHVIEN.DAT để lưu thông tin của một lớp sinh viên. Mỗi sinh viên cần những thông tin sau: Họ tên, Ngày sinh, Quê quán, Điểm trung bình, Xếp loại (trường xếp loại do chương trình tự tính lấy dựa vào điểm trung bình như sau: nếu điểm trung bình < 5 thì xếp loại 'D', nếu 5 <= điểm trung bình < 6.5 thì xếp loai 'C', nếu 6.5 <= điểm trung bình < 8 thì xếp loại 'B', trường hợp còn lại xếp loại 'A').

```
Program Vi_du_1;
Type
   St20 = String[20];
   St10 = String[10];
   SinhVien = record
            Hoten: St20;
            Ngaysinh,Quequan: St10;
            DiemTb: real;
            Xeploai: Char;
        end;
Var
  f: File of SinhVien;
  filename:String;
  Sv: sinhvien;
  Bhoten:st20;
  i:word;
Begin
    write('Nhap ten file: ');
    readln(filename);
    assign(f,filename);
    rewrite(f);
    i:=1;
    repeat
        writeln('Nhap thong tin cua cac sinh vien');
        writeln('Thong tin cua sinh vien thu ', i);
        write('Ho ten: ');
        readln(Bhoten);
        if Bhoten <> '' then
          begin
```

```
            sv.hoten:= Bhoten;
            write('Ngay sinh (dd/mm/yyyy): ');
            readln(sv.ngaysinh);
            write('Quequan: ');
            readln(sv.quequan);
            write('Diem trung binh: ');
            readln(sv.diemtb);
            if sv.diemtb<5 then
              sv.xeploai:='D'
            else
               if sv.diemtb<6.5 then
                 sv.xeploai:='C'
               else
                  if sv.diemtb<8 then
                    sv.xeploai:='B'
                  else
                     sv.xeploai:='A';
            write(f,sv);
          end;
        inc(i);
    until Bhoten = '';
    close(f);
end.
```

Bài tập 8.2: In toàn bộ nội dung của file SINHVIEN.DAT ra màn hình, nếu có, ngược lại thì thông báo "File khong ton tai".

```
Program Vi_du_2;
Type
   St20 = String[20];
   St10 = String[10];
   SinhVien = record
                Hoten: St20;
                Ngaysinh,Quequan: St10;
                DiemTb: real;
                Xeploai: Char;
              end;
Var
  f: File of SinhVien;
  Sv: sinhvien;
  Bhoten:st20;
  i:word;
Begin
    assign(f,'Sinhvien.dat');
    {$I-}
    reset(f);
    {$I+}
    if IOResult <> 0 then
      Begin
          writeln('File khong ton tai');
          exit;
      End;
```

```
    writeln(#32:10, 'DANH SACH SINH VIEN');
    writeln(#32:6,'HO TEN',#32:8,'NGAY SINH',#32:4,'QUE QUAN  DTB');
    while not eof(f) do
        begin
            read(f,sv);
            with sv do
           writeln(hoten,#32:20,length(hoten),ngaysinh,#32:2,quequan,#32:10-
length(quequan),Diemtb:5:2);
        end;
    close(f);
    readln;
End.
```

Bài tập 8.3: In danh sách tất cả sinh viên có thông tin lưu trong file SINHVIEN.DAT xếp loại khá (‘B’) trở lên.

```
Program Vi_du_3;
Type
   St20 = String[20];
   St10 = String[10];
   SinhVien = record
                 Hoten: St20;
                 Ngaysinh,Quequan: St10;
                 DiemTb: real;
                 Xeploai: Char;
              end;
Var
  f: File of SinhVien;
  filename:String;
  Sv: sinhvien;
  Bhoten:st20;
  n:word;
Begin
    assign(f,'sinhvien.dat');
    {$I-}
    reset(f);
    {$I+}
    if IOResult <>0 then
      begin
          writeln('File khong ton tai');
          exit;
      end;
    n:=0;
    writeln('Danh sach sinh vien dat loai kha tro len');
    while not Eof(f) do
        begin
            read(f,sv);
            with sv do
                if xeploai <= 'B' then { (xeploai = ‘B’) or (xeploai = ‘A’) }
                  begin
                      writeln(hoten,ngaysinh,quequan,diemtb);
                      inc(n);
```

```
                    end;
          end;
    close(f);
         writeln('Danh sach nay gom ',n,' sinh vien');
     readln;
end.
```

Bài tập 8.4: Thông tin về điểm của sinh viên có họ tên là Bhoten, ngày sinh là Bngay và quê quán là Bquequan bị sai lệch. Hãy sữa điểm và xếp loại của sinh viên này với dữ liệu nhập từ bàn phím.

```
Program Vi_du_4;
Type
    St20 = String[20];
    St10 = String[10];
    SinhVien = record
                  Hoten: St20;
                  Ngaysinh,Quequan: St10;
                  DiemTb: real;
                  Xeploai: Char;
               end;
Var
   f: File of SinhVien;
   filename:String;
   Sv: sinhvien;
   Bhoten:st20;
   Bngaysinh,Bquequan:St10;
Begin
     assign(f,'sinhvien.dat');
     {$I-}
     reset(f);
     {$I+}
     if IOResult <>0 then
        begin
            writeln('File khong ton tai');
            exit;
        end;
     write('Ho ten sinh vien: ');
     readln(bhoten);
     write('Ngay sinh: ');
     readln(Bngaysinh);
     write('Que quan: ');
     readln(bquequan);
     while not Eof(f) do
          begin
              read(f,sv);
              with sv do
                   if (hoten=bhoten) and ((ngaysinh=bngaysinh) and (quequan=bquequan))  then
                 begin
                    write('Nhap dtb can sua: ');
                    readln(diemtb);
                    if diemtb <5 then
                       xeploai:='D'
```

```
                else
                   if diemtb <6.5 then
                      xeploai:='C'
                   else
                      if diemtb <8 then
                         xeploai:='B'
                      else
                         xeploai:='A';
                n:=filepos(f);
                seek(f,n-1);
                write(f,sv);
                exit;
             end;
       end;
   Close(f);
       readln;
End.
```

Bài tập 8.5: In ra màn hình toàn bộ nội dung của một file văn bản, tên file được được nhập từ bàn phím khi thực hiện chương trình.

```
Program Vidu_5;
Var
   f: Text;
   filename,St: String;
Begin
   write('Nhap ten file: ');
   readln(filename);
   assign(f,filename);
   {$I-}
   reaset(f);
   {$I+}
   if IOResult <> 0 then
      begin
         writeln('File khong ton tai');
         halt;
      end;
   writeln('Noi dung cua file ',filename)
   while not Eof(f) do
      begin
         readln(f,st);
         writeln(st);
      end;
   close(f);
   readln;
End.
```

Bài tập 8.6: Đếm số dòng, số ký tự trắng xuất hiện trong một file văn bản đã có trên đĩa, tên file được nhập từ bàn phím khi chạy chương trình.

```
Program Vidu_6;
Var
   f: Text;
   filename,St: String;
```

```
    NLines,NStr: word;
    i: byte;
Begin
    write('Nhap ten file: ');
    readln(filename);
    assign(f,filename);
    reaset(f);
    NBl:=0;
    NStr:=0;
    while not Eof(f) do
        begin
            readln(f,st);
            inc(NStr);
            for i:= 1 to length(St) do
                if St[i] = #32 then
                    inc(NBl);
        end;
    Close(f);
          writeln('So dong : ',NStr);
    writeln('So ky tu trang: ', NBl)
    readln;
End.
```

Bài tập 8.7: Sao chép nội dung của file SINHVIEN.DAT vào file văn bản SINHVIEN.TXT sao cho mỗi sinh viên lưu trong một dòng.

```
Program Vidu_7;
Type
    St20 = String[20];
    St10 = String[10];
    SinhVien = record
                    Hoten: St20;
                    Ngaysinh,Quequan: St10;
                    DiemTb: real;
                    Xeploai: Char;
                end;
Var
    f: File of SinhVien;
    g:Text;
    St:String;
    Sv: sinhvien;
    Bdiem: String[5];
Begin
        assign(f,'sinhvien.dat');
        {$I-}
        reset(f);
        {$I+}
        if IOResult <>0 then
            begin
                writeln('File khong ton tai');
                exit;
            end;
```

```
    rewrite(g);
    while not Eof(f) do
       begin
          read(f, Sv);
          with Sv do
             begin
                Str(diemtb,bdiem:5:2);
                St:= hoten+#32+ngaysinh+#32+quequan+#32+Bdiem;
                writeln(g,St);
             end;
       end;
    Close(f);
    Close(g);
    readln;
End.
```

Bài tập 8.8: Một ma trận mxn số thực được chứa trong một file văn bản có tên MT.INP gồm: dòng đầu chứa hai số m, n; m dòng tiếp theo lần lượt chứa m hàng của ma trận. Hãy viết chương trình đọc dữ liệu từ file MT.INP, tính tổng của từng hàng ma trận và ghi lên file văn bản có tên KQ.OUT trong đó, dòng đầu chứa số m, dòng thứ hai chứa m tổng của m hàng (m,n<=200).

```
MT.INP          KQ.OUT
5 4               5
3 8 –1 5          15  4  8  12  12
5 7 –8 0
4 –3 1 6
2 4 –1 7
3 6 8 -5
Program Vidu_8;
Var
   f,g: Text;
   S:array[byte] of real;
   m,n,i,j: byte;
Begin
   assign(f,'MT.INP');
   reset(f);
   readln(f,m,n);
   fillchar(S,m,0);
   for i:= 1 to m do
      begin
         for j:=1 to n do
            begin
               read(f,x);
               S[i]:=S[i]+x;
            end;
         readln(f);
      end;
   close(f);
   assign(g,'KQ.OUT');
       rewrite(g);
   writeln(g,m);
   for i:= 1 to m do
```

```
    write(g,S[i]:0:2,#32);
  close(g);
End.
```

Chú ý:

- Chương trình trên không kiểm tra sự tồn tại của file 'MT.INP', nếu cần có thể kiểm tra tương tự các ví dụ trên.
- Tổng của mỗi hàng được lưu trong mảng một chiều S (phần tử S[i] lưu tổng của hàng i)

Bài tập 8.9: Cho 3 ma trận số nguyên A = $(a_{ij})_{mxn}$, B = $(b_{jk})_{nxp}$, C = $(c_{kl})_{pxq}$, được chứa trong file MATRIX.INP gồm: dòng đầu chứa 4 số m, n, p, q. m+n+p dòng tiếp theo lần lượt chứa m hàng ma trận A, n hàng ma trận B và p hàng ma trận C. Viết chương trình đọc dữ liệu từ file MATRIX.INP và tính ma trận tích D = AxBxC rồi ghi lên file văn bản có tên MATRIX.OUT trong đó: Dòng đầu chứa m, q; m dòng tiếp theo chứa m hàng của ma trận D.

$$d_{il} = \sum_{j}\sum_{k} * b_{jk} * c_{kl}$$

```
Program Vidu_9;
Var
  f,g: Text;
  A, B, C, D:array[1..100,1..100] of integer;
  m,n,p,q,i,j,k,l,r,s: byte;

Begin
  assign(f,'MATRIX.INP');
  reset(f);
  readln(f,m,n,p,q);
  fillchar(D,mxq,0);
  for i := 1 to m do
    begin
      for j:= 1 to n do  read(f,A[i,j]);
      readln(f);
    end;
  for j:= 1 to n do
    begin
      for k:=1 to p do read(f,B[j,k]);
      readln(f);
    end;
  for k:= 1 to p do
    begin
      for l:=1 to q do read(f,C[k,l]);
      readln(f);
    end;
  close(f);
  assign(g,'MATRIX.OUT');
  rewrite(g);
  writeln(g,m,#32,q);
  for i:= 1 to m do
    begin
      for l:=1 to q do
        begin
          for j:= 1 to n do
```

```
                for k:=1 to p do
                    D[i,l] := D[i,l] + A[i,j]*B[j,k]*C[k,l];
                write(g,D[i,l], #32);
            end;
        writeln(g);
    end;
  close(g);
  readln;
End.
```

Chú ý: Công thức tính giá trị của các phần tử ma trận $D = (d_{il})_{mxq}$ như sau:

Bài tập 8.10: Một ma trận mxn số thực được chứa trong một file văn bản có tên DULIEU.INP gồm: dòng đầu chứa hai số m, n; m dòng tiếp theo lần lượt chứa m hàng của ma trận. Hãy viết chương trình đọc dữ liệu từ file DULIEU.INP, cho biết các hàng của ma trận có tổng phần tử trên hàng đó lớn nhất. Kết quả ghi lên file văn bản có tên DULIEU.OUT , trong đó dòng đầu chứa giá trị lớn nhất của tổng các phần tử trên một hàng, dòng thứ hai chứa chỉ số các hàng đạt giá trị tổng lớn nhất đó (m,n<=100).

Chẳng hạn

```
DULIEU.INP          DULIEU.OUT
6 5                 34
3 6 8 12 2          2 5 6
7 5 6 10 6
8 2 4 5 1
3 5 6 1 3
10 12 3 1 8
8 8 8 9 1
```

```
Program Vi_du_10;
Var
  f,g: Text;
  S:array[1..100] of real;
  T: Set of byte;
  GTMax: real;
      m,n,i,j: byte;
Begin
  assign(f,'DULIEU.INP');
  reset(f);
  readln(f,m,n);
  fillchar(S,m,0);
      for i:= 1 to m do
    begin
        S:=0;
          for j:=1 to n do
           begin
              read(f,x);
              S[i]:=S[i]+x;
            end;
        readln(f);
    end;
  close(f);
  T:=[1];
  GTMax:=S[1];
```

```
    for i:= 2 to m do
        if  S[i] > GtMax then
            begin
                T:=[i];
                GtMax:= S[i];
            end
        else
            if S[i] = GTMax then
                T:= T+[i];
          assign(g,'DULIEU.OUT');
          rewrite(g);
    writeln(g,GTMax:0:2);
    for i:=1 to 100 do
        if i in T then
            write(g,i,#32);
    readln;
End.
```

Chú ý:

- Chương trình trên dùng mảng S để lưu tổng giá trị các phần tử trên mỗi hàng. Cụ thể, S[i] là tổng giá trị các phần tử trên hàng thứ i của ma trận đã cho.
- Tập T , GTMax lần lượt là tập chứa các chỉ số các hàng và giá trị lớn nhất của các phần tử trên mỗi hàng tại thời điểm đang xét. Xuất phát ta xem hàng thứ nhất có tổng giá trị lớn nhất. Khi xét hàng thứ i có các trường hợp sau:
  - S[i] > GTMax: S[i] mới là tổng lớn nhất và lúc này chỉ có hàng i đạt được giá trị này
  - S[i] = GTMax: có thêm hàng i đạt giá trị lơn nhất.
  - S[i] < GTMax: không có gì thay đổi

Bài tập 8.11: Viết chương trình sao chép nội dung của một file cho trước vào file khác, tên của file nguồn và file đích được nhập từ bàn phím khi chạy chương trình.

```
Program Sao_chep_File;
const
    bufsize = 200;
var
    f,g: file;
    File_nguon, file_dich: String;
    Buf: array[1..63000] of Byte;
    No_read, Temp: integer;
Begin
    write('Nhap ten file nguon: ');
    readln(file_nguon);
    assign(f,file_nguon);
    reset(f);
    write('Nhap ten file dich: ');
    readln(file_dich);
    assign(g,file_dich);
    rewrite(g);
    Temp:= filesize(f);
    while Temp > 0 do
        begin
            if bufsize < =Temp then
                No_read:= bufsize
```

```
        else
            No_read:= Temp;
      BlockRead((f, Buf, No_read);
      BlockWrite(g,Buf, No_Read);
      Temp:=Temp – No_read;
   end;
   close(g);
End.
```

BÀI TẬP TỰ GIẢI

Bài tập 8.12: Viết chương trình đổi tên một file đã có trên đĩa.

Gợi ý:

Dùng thủ tục Rename.

Bài tập 8.13: Viết chương trình xóa một file có trên đĩa.

Gợi ý:

Dùng thủ tục Erase.

Bài tập 8.14: Viết chương trình nối 2 file văn bản đã có trên đĩa thành một file thứ 3 với tên file được nhập vào từ bàn phím.

Gợi ý:

- Mở file 1 và file 2 để đọc dữ liệu, mở file 3 để ghi dữ liệu.
- Lần lượt đọc từng phần tử trong file 1 và 2 lưu vào file 3.
- Đóng cả ba file lại.

Bài tập 8.15: Viết chương trình thực hiện các công việc sau:

1. Tạo ra 2 file số nguyên và sắp xếp chúng theo thứ tự tăng dần.
2. Hãy nối 2 file đó lại với nhau thành file thứ 3 sao cho file mới vẫn có thứ tự tăng dần.

Gợi ý:

Xem giải thuật ở bài tập 5.15.

Bài tập 8.18: Tại một cửa hàng, người ta quản lý các hoạt động MUA/BÁN trong năm bằng cùng một loại hoá đơn. Mỗi hoá đơn là một bản ghi gồm các trường:

SoHoadon (số hoá đơn); Thang (tháng mua/bán); Mahang (mã hàng mua/bán); Loai (nhận một trong hai giá trị 'M'(mua) hoặc 'B' (bán)

Như vậy căn cứ vào trường Loai ta biết đó là hoá đơn mua hay hoá đơn bán. Viết chương trình cho phép nhập vào một dãy các hoá đơn và lưu vào file có tên Hoadon.dat, quá trình nhập dừng khi SoHoadon = 0. Tính số dư trong tháng n (n được nhập từ bàn phím khi thực hiện chương trình) . Biết rằng số dư trong một tháng được tính theo công thức:

Số dư = Tổng bán - Tổng mua,

trong đó tổng bán, tổng mua lần lượt là tổng số tiền bán, mua trong tháng đó.

Yêu cầu:

- Khi nhập chú ý kiểm tra để Loai chỉ nhận một trong hai giá trị 'M' hoặc 'B' và tháng chỉ nhận giá trị từ 1 đến 12.
- Không được sử dụng mảng.

Hướng dẫn: Khai báo file lưu các hoá đơn, mỗi hoá đơn là một bản ghi như sau

```
Type
   Hoadon = record
                SoHoadon: word;
                Thang: byte;
                Mahang: string[5];
                Loai: char;
            end;
Var
```

f: file of hoadon;

Bài tập 8.19: Người ta quản lý các đầu sách của một thư viện bằng một bản ghi gồm có các trường: Masach, Tensach, Tentacgia, Nhaxb (nhà xuất bản), Namxb (năm xuất bản), SoLuong. Viết chương trình cho phép thực hiện các thao tác sau:

a. Nhập vào các đầu sách có trong thư viện và lưu vào file có tên Sach.dat, quá trình nhập dừng khi mã sách đưa vào là một xâu rỗng.
b. Duyệt và in ra tên các quyển sách được xuất bản sau năm m (m được nhập từ bàn phím khi thực hiện chương trình).
c. Bổ sung sách vào thư viện theo yêu cầu: nếu sách đã có thì chỉ tăng số lượng sách bổ sung, ngược lại thêm một đầu sách mới vào file.

Chú ý:

- Không được sử dụng mảng
- Khi nhập chú ý kiểm tra để năm xuất bản <= năm hiện tại
- Sau khi in ra danh sách các đầu sách xuất bản sau năm m, cho biết thêm danh sách đó có bao nhiêu đầu sách tất cả.

Hướng dẫn: Khai báo thư viện là một file các đầu sách, mỗi đầu sách là một bản ghi như sau

```
Type
   St5 = String[5];
   St20 = String[20];
   Dausach = Record
                Masach: St5,
                Tensach, Tentacgia, Nhaxb: St20,
                Namxb: word;
                SoLuong: byte;
             end;
Var
   f: file of DauSach;
```

Bài tập 8.20: Người ta lưu thông tin các cán bộ trong cơ quan vào file có tên CANBO.DAT, mỗi cán bộ là một bản ghi gồm các trường: STT, Hoten, Ngaysinh, Diachi, HSLuong, HSPhucap, SoDT. Hãy viết chương trình thực hiện các yêu cầu sau:

a. Nhập danh sách cán bộ và lưu vào file, quá trình nhập dừng khi họ tên nhập vào là xâu rỗng và trường STT chương trình tự gán.
b. In ra danh sách cán bộ có hệ số lương nằm trong khoảng từ x đến y, x và y là các số thực được nhập từ bàn phím khi thực hiện chương trình.
c. Sao chép thông tin các cán bộ có tuổi trên 50 vào một file khác.
d. In bảng lương của tất cả cán bộ lưu trong file CANBO.DAT ra màn hình gồm các thông tin: STT, Hoten, HSLuong, Luong, trong đó Luong được tính theo công thức Luong = (HSLuong+HSPhucap)*290000, dữ liệu in ra định dạng theo cột. Cuối bảng, in tổng lương của toàn cơ quan.
e. Sao chép nội dung của file CANBO.DAT vào file văn bản CANBO.TXT, mỗi cán bộ tương ứng một dòng.

Hướng dẫn: Khai báo mỗi cán bộ là một bản ghi như sau

```
Type
   St10 = String[10];
   St20 = String[20];
   Canbo = Record
              Hoten, Diachi: St20,
              Ngaysinh: St10; {dd/mm/yyyy}
              HSluong, HSPhucap: real;
              SoDT: St10; {Số điện thoại }
```

```
        end;
Var
   f: file of Canbo;
```

- Khi nhập ngày sinh phải kiểm tra định dạng theo yêu cầu: dd/mm/yyyy
- Tuổi của một cán bộ được tính bằng năm hiện tại trừ cho năm sinh. Năm sinh lấy từ 4 ký tự cuối cùng của ngày sinh và chuyển sang dạng số.

Bài tập 8.21: Viết chương trình nhập vào tên một file văn bản. Kiểm tra file này có tồn tại trên đĩa không? Nếu có, in nội dung của file từ dòng thứ m đến dòng thứ n, trong đó m và n là hai số nguyên dương bất kỳ được nhập từ bàn phím khi thực hiện chương trình.

Hướng dẫn: Mở file bằng thủ tục Reset, rồi chuyển con trỏ về dòng thứ m, đọc và in n dòng (hoặc cho đến hết file).

Bài tập 8.22:Giả sử trong một file văn bản trên đĩa có tên là MATRIX.TXT người ta đã lưu các số liệu về một ma trận A cấp mxn và một vector X n chiều. Cách lưu trữ như sau:

- Dòng đầu tiên chứa hai số m và n
- Dòng thứ hai chứa vector X
- m dòng tiếp theo lần lượt chứa m hàng của ma trận A
- Giữa các số trong một dòng cách nhau một ký tự trắng

Viết chương trình tính giá trị vector Y = AX và đưa kết quả ra màn hình đồng thời lưu vào cuối file MATRIX.TXT (A và X được lấy từ file MATRIX.TXT)

Yêu cầu:

Chương trình phải thiết lập các thủ tục sau

- LayDulieu(A,X,m,n) thực hiện việc đọc dữ liệu từ file MATRIX.TXT và gán cho A, X, m, n
- TinhTich(A,X,m,n,Y) thực hiện việc tính vector Y
- LuuKetqua(Y,m) thực hiện việc in vector Y ra màn hình và lưu vào cuối file MATRIX.TXT
- Thành phần thứ i của vector Y được tính theo công thức

$$Y_i = \sum_j j * X_j$$

Bài tập 8.23: Giả sử trong một file văn bản trên đĩa có tên là DANHBA.TXT lưu danh bạ điện thoại trong thành phố. Cách lưu như sau:

- Dòng đầu lưu hai số nguyên dương m và n, trong đó m là số máy điện thoại thuộc cơ quan nhà nước, còn n là số máy thuộc tư nhân.
- m dòng tiếp theo lưu thông tin lần lượt của m máy điện thoại thuộc cơ quan nhà nước, mỗi dòng ghi số điện thoại, một ký tự trắng và sau đó là tên cơ quan.
- n dòng tiếp theo nữa lưu thông tin lần lượt của n máy điện thoại tư nhân, mỗi dòng ghi số điện thoại, một ký tự trắng và sau đó là họ tên chủ điện thoại.

Viết chương trình đọc dữ liệu từ file DANHBA.TXT và in bảng danh bạ điện thoại ra màn hình theo thứ tự tăng dần của chủ máy điện thoại, các máy điện thoại thuộc cơ quan nhà nước in trước rồi đến các máy điện thoại tư nhân. Danh sách in ra theo 3 cột, cột 1 ghi số điện thoại, cột 2 ghi tên cơ quan hoặc tên chủ máy điện thoại, cột 3 ghi loại là TN (tư nhân) hoặc NN (nhà nước)

Yêu cầu:

- Khai báo kiểu bản ghi là MAYDT bao gồm 3 trường: SoDt, TenChu, Loai
- Thiết lập thủ tục LayDulieu(A,k) để đọc dữ liệu từ file DANHBA.TXT và lưu vào mảng A (mảng các MAYDT) với k là số phần tử của mảng.
- Thiết lập thủ tục SAPXEP(A,k) để sắp xếp mỗi nhóm máy điện thoại nhà nước, tư nhân theo thứ tự tăng dần của tên chủ máy điện thoại trong mảng A.
- Thiết lập thủ tục INKETQUA(A,k) để in ra màn hình danh bạ điện thoại từ mảng A.

Chương 9

DỮ LIỆU KIỂU CON TRỎ

I. KHAI BÁO

```
Type
   <Tên kiểu con trỏ> = ^ <Kiểu của biến động>;
Var
   <Tên biến>:<Tên kiểu con trỏ>;
```

Ví dụ 1:

```
Type
   TroNguyen : ^integer;
Var
   p, q: TroNguyen;
```

Sau khai báo này các biến p và q là các biến con trỏ có thể trỏ đến các biến động có kiểu integer. Chương trình sẽ cấp phát 4 byte cho mỗi biến con trỏ. Còn vùng nhớ của các biến động chưa được cấp phát.

Ví dụ 2:

```
Type
   TroSv = ^ Sinhvien;
   Sinhvien = Record
                 Hoten: String[20];
                 Diem: real;
                 Tiep: TroSv;
              End;
Var
   p: TroSv;
```

Trong ví dụ này, p là biến trỏ có thể trỏ đến các bản ghi có kiểu Sinhvien, trong bản ghi này lại có trường Tiep là một biến trỏ có thể trỏ đến biến động khác cũng có kiểu Sinhvien.

II. LÀM VIỆC VỚI BIẾN ĐỘNG

2.1. Cấp phát vùng nhớ

Dùng thủ tục New theo cú pháp:

New(<biến trỏ>);

Phép gán giữa hai biến trỏ được thực hiện nếu chúng có cùng kiểu. Sau phép gán p:=q; các con trỏ p và q cùng trỏ đến một địa chỉ. Do đó mọi thay đổi của p^ cũng làm thay đổi q^. Như vậy, cần phân biệt hai phép gán p:=q và p^:=q^. Ngoài ra, các con trỏ cùng kiểu có thể được so sánh với nhau bằng các toán tử quan hệ = và <>.

Turbo Pascal cũng khai báo sẵn một con trỏ không trỏ tới một biến động nào gọi là con trỏ **Nil.** Giá trị con trỏ Nil là tương hợp với mọi kiểu con trỏ. Nil có thể được gán cho biến con trỏ để chỉ ra rằng con trỏ ấy hiện không được sử dụng. Chúng ta cũng có thể sử dụng Nil trong các phép so sánh.

2.2. Giải phóng vùng nhớ

Dùng thủ tục **Dispose(p);**

Trong đó p là một biến con trỏ. Thủ tục Dispose cho phép trả lại bộ nhớ động đã được cấp phát bởi thủ tục New.

III. DANH SÁCH ĐỘNG

3.1. Khái niệm

Chúng ta đã từng làm quen với kiểu mảng, lưu danh sách gồm nhiều thành phần có cùng kiểu. Mỗi thành phần là một biến tĩnh và số lượng thành phần của danh sách là cố định. Ở đây chúng ta đề cập đến một dạng danh sách động theo nghĩa: mỗi thành phần là một biến động và số lượng thành phần của danh sách có thể thay đổi. Mỗi biến động trong danh sách được gọi là một nút.

3.2. Khai báo

Để khai báo một danh sách động trước hết ta khai báo kiểu của mỗi nút trong danh sách.

```
Type <Trỏ nút> = ^ <Nút>;
     <Nút> = Record
                Data: DataType;
                Next: <Trỏ Nút>;
             End;
Var  First: <Trỏ Nút>;
```

First là địa chỉ của nút đầu tiên trong danh sách, dựa vào trường Tiep của nút này ta bết được địa chỉ của nút thứ hai, cứ như vậy ta biết được địa chỉ của tất cả các nút trong danh sách. Danh sách dạng này được gọi là danh sách liên kết đơn.

3.3. Các thao tác thường gặp trên danh sách liên kết đơn

Trong phần này chúng ta giả thiết rằng mỗi nút trong danh sách có hai trường: trường Info (lưu nội dung của biến động) và trường Next (lưu địa chỉ của nút tiếp theo). ta có khai báo danh sách như sau

```
Type TroNut = ^Nut;
Nut = Record
        Info: data; {data là kiểu dữ liệu đã định nghĩa trước}
        Next: TroNut;
      End;
Var First:TroNut;
```

3.3.1. Khởi tạo danh sách

```
First:=Nil;
```

3.3.2. Bổ sung một nút vào đầu danh sách

```
{1. Tạo ra nút mới}
New(p);
p^.Info:=X;
{2. Bổ sung vào đầu danh sách}
p^.Next:=First;
First:=p;
```

3.3.3. Bổ sung một nút vào cuối danh sách

Xuất phát danh sách không có nút nào cả. Nút mới thêm vào sẽ nằm cuối danh sách. Khi đó ta cần hai biến con trỏ First và Last lần lượt trỏ đến các nút đầu và cuối danh sách.

```
Procedure Khoitao;
var p: TroNut;
Begin
   First:= nil; Last:= nil;
   While <còn thêm nút mới vào danh sách> do
      Begin
         New(p);
         Readln(p^.Info);
         p^.Next:= Nil;
         If First = Nil then
                First:= p
         Else
                Last^.next:= p;
         Last:= p;
      End;
```

End;

3.3.4. Duyệt danh sách

Duyệt danh sách là thăm và xử lý từng nút trong danh sách.

```
Procedure Duyet;
Var p: Tronut;
Begin
  p:= First;
  While p <> nil do
     Begin
        <Xử lý p>;
        p:= p^.Next; {duyệt qua nút tiếp theo}
     End;
End;
```

3.3.5. Bổ sung một nút vào sau nút được trỏ bởi p

Thủ tục sau thực hiện việc bổ sung một nút có nội dung x vào sau nút được trỏ bởi p.

```
Procedure Bosung(p,x);
Var   q: TroNut;
Begin
   New(q);
   q^.info:=x;
   if first = nil then
      begin
         q^.next := nil;
        first := q;
      end
   else
      begin
         q^.next:= p^.next;
         p^.next:= q;
      end;
End;
```

3.3.6. Xoá một nút khỏi danh sách

Thủ tục sau thực hiện việc xóa một nút trỏ bởi p ra khỏi danh sách.

```
Procedure Xoa(p);
Var   q: TroNut;
Begin
   if First = nil then
      exit;
   if p = First then
      First := First^.next
   else
      begin
         q:= First;
         While q^.next <> p do
            q:= q^.next;
         q^.next:= p^.next;
      end;
   Dispose(p);
End;
```

BÀI TẬP MẪU

Bài tập 9.1: Trong các bài tập từ 9.1 đến 9.4, dùng danh sách liên kết đơn lưu một dãy số nguyên. Nút đầu tiên trong danh sách được trỏ bởi First. Cho khai báo mỗi nút trong danh sách như sau:

```
Type TroNut = ^ Nut;
   Nfut = Record
      GiaTri: Integer;
      Tiep: TroNut;
      End;
Var   First: TroNut;
```

Viết chương trình thực hiện các yêu cầu sau:

a. Nhập dãy các số nguyên và lưu vào danh sách có nút đầu trỏ bởi First, quá trình nhập dừng khi dữ liệu đưa vào không phải là số nguyên.

b. In giá trị các nút lớn hơn 0.

```
Program Vi_du_1;
Type TroNut = ^ Nut;
   Nut = Record
      GiaTri: Integer;
      Tiep: TroNut;
      End;
Var   First: TroNut;
   p: pointer;

Procedure Nhap;
Var
   n:integer;
   kq:boolean;
   last,p: tronut;
begin
   first:=nil;
   last:= nil;
   repeat
      write('Nhap gia tri mot nut – Ket thuc bang ky tu Q: ');
      {$I-}
      readln(n);
      {$I+}
      kq:= IOResult=0;
      if kq then
         begin
            new(p);
            p^.Giatri:=n;
            p^.Tiep:=nil;
            if first = nil then
               first:= p;
            else
               last^.Tiep:= p;
            last:=p;
         end;
   until not kq;
end;
```

```
Procedure In_so_duong;
Var
   p: Tronut;
begin
   p:= first;
   while p <> nil do
      begin
         if p^.Giatri > 0 then
            write(p^.Giatri:8);
         p:=p^.Tiep;
      end;
end;
Begin
   Mark(p);
   Nhap;
   In_so_duong;
   Release(p);
   Readln;
End.
```

Bài tập 9.2: Viết thủ tục đếm số nút có giá trị lớn hơn 0 và tính giá trị trung bình cộng của các nút đó.

```
Procedure Nut_duong(var dem: word; tb:real);
Var
   p: Tronut;
   tong:longint;
begin
   dem:=0;
   tong:=0;
   p:= first;
   while p <> nil do
      begin
         if p^.Giatri > 0 then
            begin
               inc(dem);
               tong:=tong+p^.Giatri;
            end;
         p:=p^.tiep;
   if dem = 0 then
      tb:=0
   else
      tb:= tong /dem;
end;
```

Bài tập 9.3: Giả sử dãy giá trị các nút trong danh sách đã được sắp tăng dần. Viết các thủ tục và hàm sau:

a. Procedure Insert(var first: TroNut; m: integer) thực hiện việc bổ sung một nút vào danh sách sao cho tính tăng dần được bảo toàn.

```
Procedure Insert(var first: TroNut; m: integer);
Var
   p,q: Tronut;
begin
```

```
    new(p);
    p^.Giatri:= m;
    if (first = nil) or (first^.Giatri < m ) then
        begin
            p^.Tiep:=nil;
            first:= p;
        end
    else
        begin
            q:= first;
            while (q^.Tiep <> nil) and ((q^.Tiep)^.Giatri < m) do
                q:= q^.Tiep;
            p^.Tiep:= q^.tiep;
            q^.Tiep:= p;
        end;
end;
```

b. Procedure InitList thực hiện việc tạo danh sách có tính chất trên bằng cách nhập dữ liệu từ bàn phím và quá trinh nhập dừng khi nhấn phím ESC (yêu cầu: sử dụng thủ tục Insert).

```
Procedure InitList;
Var
    m: integer;
Begin
    first:= nil;
    repeat
        write('Nhap gia tri cua mot nut: ');
        readln(m);
        insert(first,m);
    until readkey = #27;
end;
```

c. Procedure List(First: TroNut) in dãy giá trị các nút trong danh sách.

```
Procedure List(First: Tronut);
Var
    p:Tronut;
begin
    p:= first;
    while p <> nil do
        begin
            write(p^.Giatri);
            p:=p^.Tiep;
        end;
end;
```

d. Procedure DeleteZero(Var First: TroNut) thực hiện việc xoá tất cả các nút có giá trị 0 trong danh sách.

```
Procedure DeleteZero(Var First: TroNut);
var
    p,q: Tronut;
begin
    p:= first;
    while (p <> nil) and (p^.Giatri < 0) do
```

```
        begin
            q:= p;
             p:= p^.Tiep;
        end;
    while (p <> nil) and (p^.Giatri = 0) do
        begin
            q^.Tiep:= p^.Tiep;
            dispose(p);
            p:= q^.Tiep;
        end;
end;
```

e. Function TroMax(First: TroNut): TroNut trả về địa chỉ của nút đầu tiên đạt giá trị lớn nhất (tính từ đầu danh sách, nếu có, ngược lại hàm trả về giá trị Nil).

```
Function Tromax(First: TroNut);
var
    p.q: Tronut;
    m:integer;
begin
    if first = nil then
        TroMax:= nil
    else
        begin
            p:= first;
             m:= p^.Giatri;
            q:= p^.Tiep;
            while (q <> nil) do
                begin
                  if q^.Giatri > m then
                    begin
                        p:= q;
                        m:= p^.Giatri;
                    end;
                  q:= q^.Tiep;
                end;
            TroMax:=p;
            end;
end;
```

Bài tập 9.4: Giả sử danh sách khác rỗng. Viết các thủ tục và hàm sau:

a. Function GiaTriMax(First: TroNut): integer trả về giá trị lớn nhất của nút có trong danh sách.

```
Function GiaTriMax(First: TroNut): integer;
var
    m: integer;
    p, q: Tronut;
begin
    p:= first;
    m:= p^.Giatri;
    q:= p^.Tiep;
    while q<> nil do
        begin
```

```
        if q^.Giatri > m then
            m:=q^.Giatri;
        q:= q^.Tiep;
    GiaTriMax:= m;
end;
```

b. Function GiaTriMin(First: TroNut): Integer trả về giá trị nhỏ nhất của nút có trong danh sách.

```
Function GiaTriMax(First: TroNut): integer;
var
    m: integer;
    p,q: Tronut;
begin
    p:= first;
    m:= p^.Giatri;
    q:= p^.Tiep;
    while q<> nil do
        begin
            if q^.Giatri < m then
                m:=q^.Giatri;
            q:= q^.Tiep;
    GiaTriMin:= m;
end;
```

Bài tập 9.5: Cho danh sách liên kết đơn có nút đầu trỏ bởi First, được khai báo như sau

```
Type
        TroNut = ^nut;
        Nut = Record
                Info: real;
                Next: TroNut;
            End;
Var
        First: Tronut;
```

Viết các thủ tục và hàm sau:

a. Function Search(First: TroNut; k: word): TroNut trả về địa chỉ của nút thứ k (nếu có, ngược lại, hàm trả về giá trị Nil).

```
Function Search(First: TroNut; k: word): Tronut;
Var
        d: word;
        p: Tronut;
Begin
        d:=0;
        p:=first;
        while (p <> nil) do
          begin
            inc(d);
            if d = k then
                break;
            p:= p^.next;
        end;
        Search:= p;
End;
```

b. Procedure Delete_K(Var First: TroNut; k: word) thực hiện việc xoá nút thứ k trong danh sách (nếu có).

```
Procedure Delete_K(Var first: Tronut; k:word);
var
        d: word;
        p,q: Tronut;
begin
        d:=1;
        p:= first;
        while (p<> nil) and (d <k) do
            begin
               q:= p;
               p:= p^.Next;
               inc(d);
            end;
        if p <> nil then
            begin
               if  p = first then
                  first:= first^.next
               else
                  q^.next:= p^.next;
               dispose(p);
            end;
end;
```

c. Procedure DeleteList thực hiện việc xoá tất cả các nút trong danh sách.

```
Procedure DeleteList;
var
   p: Tronut;
begin
   while first <> nil do
      begin
         p:= first;
         first:= first^.next;
         dispose(p);
      end;
end;
```

Bài tập 9.6: Cho file văn bản có tên NGUYEN.INP lưu các số nguyên, giữa các số trong file cách nhau một ký tự trắng hoặc dấu xuống dòng. Viết chương trình thực hiện các yêu cầu sau:

a. Lấy dữ liệu từ file NGUYEN.INP và lưu vào danh sách liên kết đơn có nút đầu trỏ bởi First.

b. Tính tổng giá trị các nút, tổng giá trị các nút dương, tổng giá trị các nút âm, số nút có giá trị âm, số nút có giá trị dương. Các kết quả tính được sẽ lưu vào file văn bản có tên KETQUA.OUT, dòng đầu chứa 3 giá tri tổng, dòng thứ hai chứa hai giá trị còn lại.

```
Program Vi_du_6;
type
   Contro = ^ Nut;
   Nut = Record
         info: integer;
         next: Contro;
      end;
var
```

```
    first: Contro;
Procedure Lay_du_lieu;
var
    p: Contro;
    so: integer;
    f: text;
Begin
    assign(f, 'NGUYEN.INP');
    reset(f);
    first:= nil;
    while not Eof(f) do
        begin
            read(f, so);
            new(p);
            p^.info:= so;
            p^.next:= first;
            first:= p;
        end;
    close(f);
End;
Procedure Tinh_toan;
var
    f:text;
    p: Contro;
    T, T_duong, T_am: longint;
    N_duong, N_am: word;
begin
    assign(f,'KETQUA.OUT');
    rewrite(f);
    p:= first;
    T:= 0;
    T_duong: =0;
    T_am:= 0;
    N_duong:= 0;
    N_am:= 0;
    while p <> nil do
        begin
            T:= T + p^.info;
            if p^.info > 0 then
                begin
                    T_duong:= T_duong + p^.info;
                    inc(N_duong);
                end;
            if p^.info < 0 then
                begin
                    T_am:= T_am + p^.info;
                    inc(N_am);
                end;
            p:= p^.next;
        end;
```

```
    writeln(f, T,#32,T_duong,#32,T_am);
    writeln(f,N_duong,#32,N_am);
    close(f);
end;
Begin
    Lay_du_lieu;
    Tinh_toan;
End.
```

Bài tập 9.7: Người ta lưu thông tin các bệnh nhân của bệnh viện X trong danh sách liên kết đơn có nút đầu trỏ bởi First, mỗi bệnh nhân tương ứng với một nút trong danh sách được khai báo như sau:

```
Type St20 = String[20];
         St5 = String[5];
         St2 = String[2];
    TroBN = ^BenhNhan;
         BenhNhan = Record
              MaBN: St5;        {Mã bệnh nhân}
                                Hoten: St20;       {Họ tên bệnh nhân}
              Tuoi: byte;       {Tuổi}
              Tiep: TroBN;
           End;
```

Chú ý: *Hai ký tự đầu của mã bệnh nhân là mã của khoa điều trị.*

Viết các thủ tục và hàm sau:

a. **Procedure BoSungBN(Var First: TroBN; Bma: St5; Bten: St20; Btuoi: byte)** bổ sung bệnh nhân có mã là Bma, họ tên là Bten, tuổi là Btuoi vào cuối danh sách có nút đầu trỏ bởi First (Lưu ý: Kiểm tra Bma chưa có trong danh sách mới bổ sung).

```
Procedure BoSungBN(var First: TroBN; Bma: St5; Bten: St20; Btuoi:byte);
var
    p,q: TroBN;
begin
    p:= first;
    while (p <> nil) and (p^.MaBN <> Bma) do
        begin
            q:= p;
            p:= p^.tiep;
        end;
    if p = nil then
        begin
            new(p);
            p^.MaBn:= Bma;
            p^.Hoten:= Bten;
            p^.tuoi:= Btuoi;
            p^.Tiep:= nil;
            if first = nil then
                first:= p
            else
                q^.tiep:= p;
end;
```

b. Procedure KhoiTao(Var First: TroBN) nhập dữ liệu cho danh sách có nút đầu trỏ bởi First, quá trình nhập dừng khi mã bệnh nhân đưa vào là xâu rỗng (Yêu cầu sử dụng thủ tục BoSungBN).

```
Procedure KhoiTao(Var First: TroBN);
var
   bma:St5;
   bten: st20;
   btuoi: byte;
begin
   first:= nil;
   repeat
      write('Nhap ma benh nhan: ');
         readln(bma);
      if bma <> '' then
         begin
            write('Ho ten benh nhan: ');
            readln(bten);
               write('Tuoi: ');
            readln(btuoi);
            BosungBN(first, bma, bten, btuoi);
         end;
   until bma = '';
end;
```

c. Function SoBN(First: TroBN; BKhoa: St2): word trả về số lương bệnh nhân điều trị tại khoa có mã BKhoa.

```
Function SoBN(First: TroBN; BKhoa: St2): word;
Var
   p: TroBN;
   dem:word;
Begin
   dem:= 0;
   p:= first;
   while p <> nil do
      begin
         if copy(p^.MaBN,1,2) = BKhoa then  inc(dem);
         p:= p^.tiep;
      end;
   SoBN:= dem;
End;
```

d. Procedure LietKe(First: TroBN; n: byte) in thông tin của các bệnh nhân có tuổi nhỏ hơn hoặc bằng n.

```
Procedure LietKe(First: TroBN; n: byte);
Var
   p: TroBN;
Begin
   p:= first;
   while p <> nil do
      begin
```

```
        with p do
            if tuoi <= n then
                writeln(mabn, #32,hoten, #32, tuoi);
            p:= p^.tiep;
    end;
End;
```

e. Procedure XoaBN(Var First: TroBN; Bma: St5) xoá bệnh nhân có mã Bma khỏi danh sách.

```
Procedure XoaBN(Var First: TroBN; Bma: St5);
Var
   p,q: TroBN;
Begin
   p:= first;
   while (p <> nil) and (p^.MaBN <> Bma) do
      begin
         q:= p;
         p:= p^.tiep;
      end;
   if p <> nil then
      begin
         if p = first then
            first:= first^.tiep
         else
            q^.tiep:= p^.tiep;
         dispose(p);
End;
```

Bài tập 9.8: Người ta lưu thông tin của mỗi đại lý trong công ty bởi một nút trong danh sách liên kết đơn và được khai báo như sau:

```
Type St6 = String[6];
   TroDL = ^ DaiLy;
   DaiLy = Record
         SoDT: St6;
         DoanhThu: LongInt;
         Next: TroDL;
      End;
```

Viết các thủ tục và hàm:

a. Procedure BoSung(Var First: TroDL; Tel: St6; m: LongInt) để bổ sung một đại lý có số điện thoại Tel và doanh thu là m vào đầu danh sách có nút đầu trỏ bởi First.

```
Procedure BoSung(Var First: TroDL; Tel: St6; m: LongInt);
Var
   p: TroDL;
Begin
   new(p);
   p^.SoDt:= Tel;
   p^.Doanhthu:= m;
   p^.next:= first;
   first:= p;
End;
```

b. Function DThu(First: TroDL; Tel: St6): LongInt trả về doanh thu của đại lý có số điện thoại là Tel, nếu không có đại lý đó thì hàm trả về giá trị 0.

```
Function DThu(First: TroDL; Tel: St6): LongInt;
Var
   p: TroDL;
Begin
   p:= first;
   while (p <> nil) and (p^.SoDT <> Tel) do
      p:= p^.next;
   if p <> nil then
      Dthu:= p^.doanhthu
   else
      Dthu:= 0;
End;
```

c. Function TongDThu(First: TroDL): Real trả về tổng doanh thu của tất cả các đại lý trong công ty.

```
Function TongDThu(First: TroDL): Real;
Var
   p: TroDl;
   T: real;
Begin
   T:= 0;
   p:= first;
   while p <> nil do
      begin
         T:= T+ p^.Doanhthu;
         p:= p^.next;
      end;
   TongDthu:= T;
End;
```

d. Function DemDL(First: TroDL; m: LongInt): Word trả về số đại lý của công ty có doanh thu lớn hơn m.

```
Function DemDL(First: TroDL; m: LongInt): Word;
Var
   p: TroDL;
   dem: word;
Begin
   dem:= 0;
p:= first;
   while p <> nil do
      begin
         if p^.Doanhthu > m then
            inc(dem);
         p:= p^.next;
      end;
   DemDL:= dem;
End;
```

e. Procedure XoaDL(Var First: TroDL; Tel: St6) xóa đại lý có số điện thoại Tel ra khỏi danh sách.

```
Procedure XoaDL(Var First: TroDL; Tel: St6);
Var
   p,q: TroDL;
Begin
   p:= first;
   while (p <> nil) and (p^.SoDT <> Tel) do
      begin
         q:= p;
         p:= p^.next;
      end;
   if p <> nil then
      begin
         if p = first then
            first:= first^.next
         else
            q^.next:= p^.next;
         dispose(p);
      end;
End;
```

BÀI TẬP TỰ GIẢI

Bài tập 9.9: Dùng danh sách móc nối để biểu diễn một đa thức $P_n(x) = a_n x^n + a_{n-1}x^{n-1} +...+ a_0$. Trong đó, mỗi số hạng của đa thức được xác định bởi 2 thành phần: hệ số a_i và số mũ i.

Như vậy, ta có thể xây dựng cấu trúc dữ liệu cho đa thức như sau:

```
TYPE    DATHUC = ^SOHANG;
        SOHANG = Record
           HeSo: Real;
           SoMu: Integer;
           Next: DATHUC;
        End;
```

Viết chương trình thực hiện các công việc sau:

1. Viết thủ tục nhập vào một đa thức.
2. Viết thủ tục để sắp xếp lại các số hạng của đa thức theo thứ tự số mũ giảm dần.
3. Viết thủ tục/hàm để cộng 2 đa thức.
4. Viết hàm để tính giá trị của đa thức theo giá trị X.

Bài tập 9.10: Cho một file văn bản trong đó có chứa các từ. Các dấu phân cách từ là: ký tự trắng, dấu chấm, dấu phẩy, dấu chấm phẩy, dấu hai chấm, dấu than, dấu hỏi. Mọi từ đều bắt đầu bằng một ký tự trong tập ['A'..'Z'].

1. Viết thủ tục cho phép đọc các từ trong file văn bản đã cho và lưu các từ đó vào mảng các danh sách móc nối:

```
TuDien : ARRAY['A'..'Z'] OF DanhSach;
```

Trong đó kiểu DanhSach được cho như sau:

```
TYPE DanhSach = RECORD
                   Tu : String[10];
                   Next : DanhSach;
                END;
```

Mỗi danh sách móc nối trong từ điển đều phải được sắp thứ tự (tăng dần), và các từ được lưu trong từ điển phải khác nhau.

2. Viết một thủ tục hiển thị tất cả các từ trong từ điển ra màn hình theo thứ tự tăng dần.

3. Viết một thủ tục bổ sung một từ mới vào từ điển bằng cách đọc từ đó từ bàn phím, tìm nó trong từ điển.

- Nếu tìm thấy, hiển thị thông báo:"Từ đã có trong từ điển".
- Nếu không tìm thấy, chèn từ đó vào trong từ điển ở vị trí thích hợp.

Bài tập 9.11: Cho dãy số nguyên sắp theo thứ tự tăng dần và lưu trong một danh sách liên kết đơn có địa chỉ nút đầu danh sách là First.

a. Viết chương trình xoá tất cả các nút trong danh sách có giá trị 0.

b. Viết chương trình in ra các giá trị phân biệt của danh sách.

Bài tập 9.12: Cho hai dãy số thực lưu trong hai danh sách liên kết đơn, có địa chỉ của các nút đầu danh sách lần lượt là First1 và First2. Giả sử trong mỗi danh sách giá trị các nút đã được sắp tăng dần. Hãy viết chương trình tạo một danh sách liên kết đơn có nút đầu trỏ bởi List, chứa tất cả các phần tử của hai danh sách trên, danh sách mới này cũng được sắp thứ tự.

Bài tập 9.13: Một công ty du lịch quản lý tất cả các xe ô tô của họ bằng một danh sách liên kết, mỗi nút của danh sách được khai báo như sau:

```
Type
   TroXe = ^Xe;
   St6 = String[6];
   St20 = String[20];
   Xe = Record
      TaiXe: St20; { họ tên tài xế }
      BienSo: St6; { biển số xe }
      Socho: Byte;   { số chỗ ngồi }
      Tiep: TroXe;
       end;
Var
   First: TroXe;
```

a. Viết thủ tục Procedure Print(First: TroXe; n:byte); in họ tên tài xế, biển số xe của tất cả các xe có n chỗ ngồi được lưu trong danh sách.

b. Viết hàm Function SoChoNgoi(First: TroXe; Bso: St6); trả về số chỗ của xe có biển số Bso.

Bài tập 9.14: Người ta quản lý các sách trong thư viện bằng một danh sách liên kết, sắp theo thứ tự của mã sách. Mỗi đầu sách tương ứng với một nút trong danh sách có khai báo như sau:

```
Type
   TroSach = ^Sach;
   St4 = String[4];
   St20 = String[20];
   Sach = Record
          Ma: St4;
          Ten, Tacgia: St20;
          NamXb: word;
          Soluong: Byte;
          Next: TroSach;
       end;
Var
   First: TroSach;
```

Chú ý: Các đầu sách được săp theo thứ tự mã sách.

a. Viết thủ tục Procedure CapNhat(Var First: TroSach; Bma:St4; Bten, BTgia: St20; Bnam: word; n: byte); bổ sung vào thư viện đầu sách có mã là Bma, tên sách Bten, tác giả BTgia và số lượng bổ sung là n theo yêu cầu: Nếu đầu sách có mã là Bma đã có trong thư viện

thì chỉ tăng số lượng lên n, ngược lại thêm một đầu sách mới vào thư viện với số lượng n và bảo toàn thứ tự của mã sách.

b. Viết thủ tục Procedure LietKeNam(First: TroSach; Nam: word) in danh sách các đầu sách xuất bản vào năm Nam.

c. Viết hàm Function So_Dau_Sach(First: TroSach; BTgia: St20) trả về số đầu sách của tác giả BTgia.

d. Viết thủ tục Procedure LietKeten(First: TroSach; Bten: St20) in danh sách tất cả các đầu sách có tên sách là Bten.

Bài tập 9.15: Một cửa hàng kinh doanh vật liệu xây dựng quản lý lượng hàng tồn kho bằng một danh sách liên kết. Mỗi loại vật liệu tương ứng với một nút trong danh sách và có khai báo như sau:

```
Type
   St3 = String[3];
   St10 = String[10];
   TroVT = ^Vattu;
   Vattu = Record
         Ma: St3;
         Ten: St10;
         DVTinh: St10;     { đơn vị tính}
         Soluong: word;
         Tiep: TroVattu;
       End;
Var
   First: TroVattu;
```

a. Viết thủ tục Procedure XuatKho(Var First: TroVattu; Bma: St3; Bdonvi: St10; n: word); lấy ra khỏi kho n bdonvi loại vật tư có mã là Bma theo yêu cầu sau:

- Nếu vật tư có mã Bma không có trong kho thì thông báo kho không có loại vật tư này và kết thúc thực hiện.
- Ngược lại, kiểm tra Bdonvi có trùng với DVTinh của loại vật tư này không, nếu không trùng thì yêu cầu đổi theo đơn vị tính lưu trong danh sách trước khi thực hiện xuất kho.
- Nếu số lượng trong kho của vật tư có mã Bma nhỏ hơn n thì đưa ra thông báo và hỏi lại có muốn xuất không, nếu muốn thì xuất với số lượng bao nhiêu?
- Sau khi xuất n đơn vị loại vật tư theo yêu cầu, giảm số lượng vật tư trong kho cho phù hợp và nếu số lượng của vật tư này bằng 0 thì xoá nó ra khỏi danh sách.

b. Viết thủ tục Procedure NhapKho(Var First: TroVattu; Bma: St3; Bten, Bdv: St10; n: word); nhập vào kho n Bdv laọi vật tư có mã Bma theo yêu cầu:

- Nếu loại vật tư có mã Bma đã có trong kho thì chỉ tăng số lượng vật tư này trong kho, nhớ kiểm tra đơn vị tính như câu a.
- Ngược lại, bổ sung một loại vật tư mới vào kho với mã là Bma, tên vật tư là Bten, đơn vị tính là Bdv và số lương tương ứng là n.

c. Viết hàm Function SoVattu(First: TroVattu; Bma:St3); trả về số lượng vật tư có mã là Bma còn tồn trong kho, nếu không có vật tư này, hàm trả về giá trị 0.

d. Viết thủ tục Procedure ThongKe(First: TroVattu); in ra thông tin của tất cả các loại vật tư hiện có trong kho.

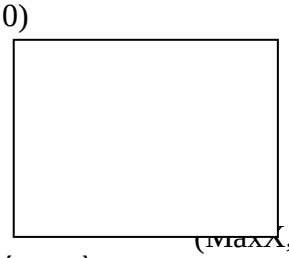
Chương 10
ĐỒ HỌA

I. MÀN HÌNH TRONG CHẾ ĐỘ ĐỒ HỌA (GRAPHIC)

Hình ảnh trong chế độ đồ họa được tạo ra bằng các điểm ảnh (Pixel), số điểm ảnh của màn hình đồ họa tùy thuộc vào từng loại CARD màn hình và MODE qui định cho màn hình đó.

Việc lập trình trong chế độ đồ họa cần phải xác định được loại màn hình đang sử dụng và chương trình phải vận hành được trên nhiều loại màn hình khác nhau.

Tọa độ của một điểm ảnh trên màn hình đồ họa cũng giống như trong chế độ văn bản (TEXT) với điểm ảnh đầu tiên trên góc trái màn hình là (0,0), tọa độ đỉnh dưới phải tùy thuộc vào độ phân giải của màn hình, CARD màn hình và MODE màn hình.



Để dử dụng được chế độ đồ họa trên màn hình, ta cần phải có các File sau:

GRAPH.TPU Chứa các lệnh đồ họa
* .BGI Chứa Font màn hình
* .CHR Chứa Font ký tư

II. KHỞI TẠO VÀ THOÁT KHỎI CHẾ ĐỘ ĐỒ HỌA

2.1. Khởi tạo chế độ đồ họa

Thủ tục **INITGRAPH**(Gd,Gm:Integer; Path:String);

trong đó:

- Gd: Chỉ CARD màn hình.

Thông thường, một chương trình phải được chạy trên nhiều loại màn hình khác nhau nên ta có thể khai báo:

Gd = Detect (= 0)

Với hằng Detect, máy sẽ tự động tìm CARD màn hình tương ứng để chạy chương trình.

- Gm: Chỉ MODE màn hình.

Trong trường hợp khai báo Gd = Detect thì không cần thiết phải khai báo Gm vì máy tính sẽ tự xác định loại CARD màn hình và thiết lập chế độ MODE màn hình tương ứng với CARD màn hình đó.

- Path: Đường dẫn đến nơi chứa các file *.BGI. Nếu Path = ‘’ thì ta hiểu là các file *.BGI nằm trong thư mục hiện hành.

Hàm **GRAPHRESULT**:Integer;

Hàm này trả về kết quả của việc khởi động đồ họa.

= 0 : Thành công.

<>0 : Bị lỗi.

Tên của lỗi được xác định bởi hàm **GRAPHERRORMSG**(Er:Integer):String;

Hàm này cho ra một xâu ký tự thông báo lỗi của đồ họa xác định bởi đối số Er.

* Hằng số GrOK = 0: Việc khởi động đồ họa có lỗi.

Ví dụ:

```
Uses Graph;
Procedure ThietLapDoHoa;
 var gd,gm,Gr:integer;
 Begin
   DetectGraph(Gd,Gm);
   InitGraph(gd,gm,'C:\TP\BGI');
```

```
Gr:=GraphResult;
  If Gr<>GrOK then
    Begin
      writeln('Loi Do hoa: ',GraphErrorMsg(Gr));
      Halt(1);
    End;
End;
BEGIN
  ThietLapDoHoa;
  . . .
END.
```

Chú ý: Ta có thể khởi tạo mode đồ hoạ với chế độ 256 màu bằng cách sử dụng hàm **InstallUserDriver**(Name:String;Ptr:Pointer):Integer; với điều kiện trên đĩa phải có file **SVGA256.BGI**.

```
Procedure ThietLapDoHoa;
 var gd,gm,Gr:integer;
 Begin
   Gd:= InstallUserDriver('  SVGA256'  ,NIL);
   Gm:=2; {Mode 640x480x256}
   InitGraph(gd,gm,'C:\TP\BGI');
 End;
```

2.2. Thoát khỏi chế độ đồ họa

Thủ tục **CLOSEGRAPH;**

Sau đây là cấu trúc chung của một chương trình đồ họa:.

```
Uses Crt,Graph;
Procedure ThietLapDoHoa;
 var gd,gm,Gr:integer;
 Begin
   DetectGraph(Gd,Gm);
   InitGraph(gd,gm,'C:\TP\BGI');
   Gr:=GraphResult;
   If Gr<>GrOK then
     Begin
       writeln('Loi Do hoa: ',GraphErrorMsg(Gr));
       Halt(1);
     End;
 End;
BEGIN
   ThietLapDoHoa;
   . . .
   CloseGraph;
END.
```

III. TỌA ĐỘ VÀ CON TRỎ TRÊN MÀN HÌNH ĐỒ HỌA

3.1. Hàm GetMaxX:Integer;

Cho tọa độ cột lớn nhất của màn hình.

3.2. Hàm GetMaxY:Integer;

Cho tọa độ dòng lớn nhất của màn hình.

3.3. Thủ tục MOVETO(x,y:Integer);

Di chuyển con trỏ từ vị trí đang đứng đến tọa độ (x,y).

3.4. Thủ tục MOVEREL(dx,dy:Integer);

Di chuyển con trỏ từ vị trí đang đứng đến tọa độ mới cách tọa độ cũ khoảng cách là dx, dy.

3.5. Vẽ một điểm trên màn hình:

Dùng thủ tục PUTPIXEL(x,y:Integer; color:Word);

3.6. Lấy màu của một điểm tại tọa độ x,y:

Hàm GETPIXEL(x,y:Integer):Word;

IV. ĐẶT MÀU TRÊN MÀN HÌNH ĐỒ HỌA

4.1. Đặt màu cho đối tượng cần vẽ

Dùng thủ tục SETCOLOR(Color:Byte);

4.2. Đặt màu nền

Dùng thủ tục SETBKCOLOR(Color:Byte);

V. CỬA SỔ TRONG CHẾ ĐỘ ĐỒ HỌA

5.1. Đặt cửa sổ trên màn hình

Thủ tục SETVIEWPORT(x1,y1,x2,y2:Integer; Clip:Boolean);

Với x1,y1: đỉnh trên trái của cửa sổ.

x2,y2: đỉnh dưới phải của cửa sổ.

Clip = TRUE: những gì vượt khỏi màn hình sẽ bị cắt bỏ.

Clip = FALSE: những gì vượt khỏi màn hình sẽ không bị cắt bỏ.

* Khi tạo cửa sổ thì tọa độ trên màn hình sẽ thay đổi theo.

Tọa độ mới = Tọa độ cũ - Tọa độ đỉnh trên trái.

5.2. Xóa hình ảnh trong cửa sổ

- Xóa hình ảnh trong cửa sổ, ta dùng thủ tục CLEARVIEWPORT;
- Xóa toàn bộ màn hình, ta dùng thủ tục CLEARDEVICE;

VI. VIẾT CHỮ TRONG CHẾ ĐỘ ĐỒ HỌA

6.1. Chọn Font chữ

Ta dùng thủ tục SETTEXTSTYLE(font,Dir,size:Word);

- Các font có thể chứa các hằng sau:

  DefaultFont = 0; TriplexFont = 1; SmallFont = 2;

  SansSerifFont = 3; GothicFont = 4;
- Dir có các hằng sau:

  HorizDir = 0 Từ trái qua phải.

  VetDir = 1 Từ dưới lên trên.
- Size: độ lớn của chữ.

6.2. Chọn phân bố chữ

Dùng thủ tục SETTEXTJUSTIFY(Hz,Vt:Word);

Chọn vị trí của chữ xung quanh tọa độ định sẵn.

- Hz là phân bố chữ theo trục ngang. Có các hằng sau:

  LeftText = 0 Chữ viết nằm bên phải trục đứng.

  CenterText = 1 Chữ viết nằm ở giữa trục đứng.

  RightText = 2 Chữ viết nằm bên trái trục đứng.
- Vt là bố trí chữ theo hướng dọc đối với tọa độ qui định xuất chuỗi. Các hằng liên quan:

  BottomText = 0 Chữ viết nằm bên trên trục ngang.

  CenterText = 1 Chữ viết nằm ở giữa trục ngang.

  TopText = 2 Chữ viết nằm bên dưới trục ngang.

6.3. Viết một xâu ký tự lên màn hình

- Xuất một xâu ký tự tại vị trí con trỏ:

  Dùng thủ tục OUTTEXT(St:String);
- Xuất một xâu ký tự tại tọa độ x,y:

  Dùng thủ tục OUTTEXTXY(x,y:Word; St:String);

Chú ý: Cách xuất chuỗi của hai thủ tục trên được qui định trong thủ tục SETTEXTJUSTIFY và SETTEXTSTYLE.

VII. VẼ CÁC HÌNH CƠ BẢN

7.1. Chọn kiểu đường

Dùng thủ tục SETLINESTYLE(Ls,Pt,Tk:Word);

Thủ tục này xác định kiểu đường được vẽ trong đồ họa.

Ls: kiểu đường vẽ. Ls có các giá trị sau:

0: Đường liền nét
1: Nét đứt
2: Nét chấm gạch
3: Nét gạch
4: Đường do người thiết kế tạo ra.

Pt: xác định màu vẽ.

. Nếu Ls = 0..3 thì Pt=0 (Lấy giá trị Default)
. Nếu Ls = 4 thì Pt là số nguyên chỉ màu của kiểu đường.

Tk: xác định độ dày của đường.

Tk = 1: bình thường.
Tk = 3: đậm nét.

7.2. Vẽ đoạn thẳng

LINE(x1,y1,x2,y2:Integer); vẽ từ điểm (x1,y1) đến điểm (x2,y2)

LINETO(x,y:Integer); vẽ từ vị trí con trỏ đến điểm (x,y)

LINEREL(dx,dy:Integer); vẽ từ vị trí con trỏ đến điểm cách đó một khoảng dx,dy.

7.3. Vẽ hình chữ nhật

Dùng thủ tục RECTANGLE(x1,y1,x2,y2:Integer);

7.4. Vẽ cung tròn

Thủ tục ARC(x,y:Integer; g1,g2,R:Word);

Vẽ cung tròn có tâm (x,y) bán kính R, góc bắt đầu là g1 và góc kết thúc là g2.

7.5. Vẽ đường tròn - Ellip

Thủ tục vẽ đường tròn: CIRCLE(x,y:Integer; R:Word);

Thủ tục ELLIPSE(x,y:integer; g1,g2,Rx,Ry:Word);

Vẽ Ellip có tâm (x,y) bán kính ngang Rx, bán kính dọc Ry, góc bắt đầu là g1 và góc kết thúc là g2.

7.6. Định MODE đường vẽ

Thủ tục SETWRITEMODE(Mode:Integer);

- Định Mode vẽ cho các đường thẳng.
- Ta có thể chọn Mode bằng các hằng:

CopyPut = 0; XORPut = 1;

Trong đó:

. CopyPut là Mode chèn, đường mới sẽ không xóa đường cũ.
. XORPut là Mode xóa, đường mới sẽ xóa đường cũ.

XIII. TÔ MÀU CÁC HÌNH

8.1. Chọn kiểu tô

Thủ tục SETFILLSTYLE(Pt,Cl:Word);

Với:

- Pt: Mẫu tô của hình. Có các hằng từ 0 đến 12.

0: Tô bằng màu nền.
1: Tô bằng màu viền.
2: Tô bằng các dấu ---
.....

- Cl: Màu tô của hình.

8.2. Vẽ hình chữ nhật có tô màu ở bên trong

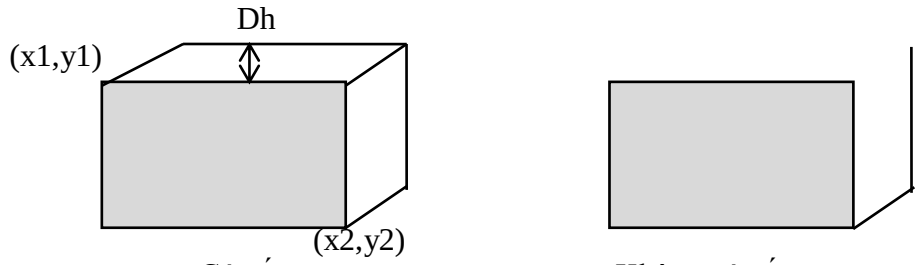
Thủ tục BAR(x1,y1,x2,y2:Integer);

Vẽ hình chữ nhật có tô màu và mẫu tô được xác định bởi thủ tục SETFILLSTYLE.

8.3. Vẽ hình hộp chữ nhật

Thủ tục BAR3D(x1,y1,x2,y2,Dh:Word; Top:Boolean);

Vẽ hình hộp chữ nhật có tọa độ đỉnh trên là (x1,y1), đỉnh dưới là (x2,y2) và chiều dày là Dh.



Có nắp Không có nắp

Top = TRUE: Hình hộp có nắp.

Top = FALSE: Hình hộp không có nắp.

8.4.Vẽ hình Ellip

Thủ tục FILLELLIPSE(x,y:Integer; Rx,Ry:Word);

8.5. Vẽ hình quạt tròn

Thủ tục PIESLICE(x,y:Integer; g1,g2,R:Word);

Vẽ hình quạt tròn có tâm (x,y), góc đầu g1, góc cuối g2, bán kính R.

8.6. Vẽ hình quạt Ellip

thủ tục SECTOR(x,y:Integer; g1,g2,Rx,Ry:Word);

8.7. Làm loang màu một vùng kín

Thủ tục FLOODFILL(x,y:Integer; Color:Word);

Trong đó:

(x,y): điểm nằm trong vùng kín.

Color: màu muốn tô.

8.8. Vẽ đa giác

Đối với một đa giác bất kỳ có N đỉnh, ta phải khai báo N+1 đỉnh để vẽ đường gấp khúc với tọa độ điểm đầu trùng với tọa độ điểm cuối.

Để vẽ đa giác ta dùng thủ tục: DRAWPOLY(Np:Word; Var P);

trong đó:

Np: số đỉnh của đa giác + 1

P: chứa tọa độ các đỉnh, là một mảng có Np thành phần có kiểu dữ liệu là PointType được định nghĩa trong Unit Graph như sau:

```
TYPE    PointType = Record
                      x,y: Integer;
                    End;
```

IX. CÁC KỸ THUẬT TẠO HÌNH CHUYỂN ĐỘNG

9.1. Kỹ thuật lật trang màn hình

CARD màn hình có nhiều trang, mỗi trang được đánh số 0,1,2,...

Để vẽ hình lên một trang màn hình, ta dùng thủ tục:

SETACTIVEPAGE(Page:Word);

Trong đó, Page là số của trang màn hình. Thủ tục này được đặt trước khi có lệnh vẽ lên màn hình.

Để đưa trang màn hình ra màn hình, ta dùng thủ tục:

SETVISUALPAGE(Page:Word);

Page: trang màn hình muốn xem.

Thông thường, màn hình sẽ làm việc và hiện ra trên trang 0. Do đó, để vừa xem màn hình vừa vẽ lên trang màn hình khác, ta thường dùng hai thủ tục trên đi kèm với nhau.

Để thực hiện tự động chương trình khi sử dụng cú pháp lật hình này, ta thường theo một giải thuật sau:

- Tạo biến page1,page2:Word;
- Tạo vòng lặp

```
...
Repeat
   SetVisualPage(page1); (* Xem trang màn hình page1 *)
   SetActivePage(page2); (* Vẽ hình lên trang page2 *)
   .......
   < Các thủ tục vẽ hình >
   .......
   (* Hoán vị 2 biến page1, page2 *)
   Temp:=page1;
   page1:=page2;
   page2:=Temp;
Until <Điều kiện thoát>;
```

9.2. Lưu và di chuyển một vùng màn hình

Chúng ta có thể lưu một vùng màn hình vào bộ nhớ rồi sau đó lại dán nó lên màn hình tại một vị trí khác.

Lưu một vùng màn hình vào bộ nhớ được thực hiện bằng thủ tục:

```
GETIMAGE(x1,y1,x2,y2:Integer; Var P:Pointer);
```

trong đó P là con trỏ để lưu nội dung của vùng (x1,y1,x2,y2).

Việc đăng ký một vùng nhớ động phải được khai báo dung lượng cần thiết. Dung lượng vùng nhớ được thực hiện bằng hàm:

```
IMAGESIZE(x1,y1,x2,y2:Integer):Word;
```

Để hiện hình ảnh từ bộ nhớ ra màn hình, ta dùng thủ tục:

```
PUTIMAGE(x,y:Integer; P:Pointer; Mode:Word);
```

trong đó:

(x,y): Tọa độ đỉnh trái hình chữ nhật mà ta muốn đưa ra.

P : Con trỏ lưu vùng hình chữ nhật.

Mode: Hằng số chỉ phương thức hiện ra màn hình. Mode chứa một trong các hằng sau:

NormalPut = 0: Xuất ra như đã lưu (phép MOV)
XORPut = 1: Phép XOR, xóa hình cũ nếu hai hình giao nhau.
ORPut = 2: Phép OR, lấy cả hai hình nếu hai hình giao nhau.
ANDPut = 3: Phép AND, nếu hai hình giao nhau thì lấy phần chung.
NOTPut = 4: Phép NOT, cho ra âm bản.

Về việc thực hiện ta tiến hành như sau:

- Khai báo một biến con trỏ P:Pointer;
- Đăng ký một vùng nhớ động do P quản lý bằng thủ tục
  GETMEM(P,ImageSize(x1,y1,x2,y2));
- Lưu hình ảnh bằng thủ tục GETIMAGE(x1,y1,x2,y2,P^);
- Xuất ra màn hình bằng thủ tục PUTIMAGE(x,y,P^,Mode);

BÀI TẬP MẪU

Bài tập 10.1: Viết dòng chữ có bóng trong chế độ 256 màu.

```
Uses crt,Graph;
var gd,gm:integer;

Procedure WriteStr(dx,dy:Integer;st:String);
```

```
 Var i,j:Integer;
 Begin
  settextstyle(5,0,8);
  j:=16;
  (* Viet chu bong *)
  for i:=0 to 15 do
    begin
      setcolor(j);
      outtextxy(dx+i,dy+i,st);
      inc(j);
    end;
  setcolor(40);
  outtextxy(dx+i,dy+i,st);
 End;

Begin
 gd:=INSTALLUSERDRIVER('SVGA256',NIL);
 GM:=4;
 initgraph(gd,gm,'c:\bp\BGI');
 WriteStr(1,100,' Pham Anh Phuong');
 readln;
 CloseGraph;
End.
```

Bài tập 10.2: Vẽ các hình chữ nhật ngẫu nhiên trên màn hình.

```
Uses Crt,Graph;
Procedure ThietLapDohoa;
Var Gd,Gm:Integer;
Begin
    Gd:=0;
    InitGraph(Gd,Gm,'D:\BP\BGI');
End;

Function RandColor:Byte;
  Begin
     RandColor:=Random(MaxColors - 1)+1;
  End;
Procedure DeMo;
   Var  x1,y1,x2,y2:Integer;
   Begin
      Randomize;
      Repeat
    x1:=Random(GetMaxX);
    y1:=Random(GetMaxY);
    x2:=Random(GetMaxX - x1) + x1;
    y2:=Random(GetMaxX - y1) + y1;
    SetColor(RandColor);
    Rectangle(x1,y1,x2,y2);
    Delay(500);
     Until KeyPressed;
   End;
```

```
BEGIN
  ThietLapDohoa;
  DeMo;
  CloseGraph;
END.
```

Bài tập 10.3: Vẽ một kim đồng hồ quay quanh tâm O(x0,y0).

```
Uses crt,Graph;
Var x0,y0:word;
     Alpha,Beta,R:real;

Procedure VeDgt(x0,y0:word; R,Alpha:real);
 Begin
   Line(x0,y0,x0+Round(R*Cos(Pi*Alpha/180)),
                y0-Round(R*Sin(Pi*Alpha/180)));
 End;
BEGIN
 ThietLapDoHoa;
 x0:=GetMaxX div 2;
 y0:=GetMaxY div 2;
 R:=100;
 Alpha:=90;
 Beta:=6;
 SetWriteMode(XORPut);
 VeDgt(x0,y0,R,Alpha);
 Repeat
 VeDgt(x0,y0,R,Alpha);
  Alpha:=Alpha - Beta;
  VeDgt(x0,y0,R,Alpha);
  Delay(250);
 Until KeyPressed;
 CloseGraph;
END.
```

Bài tập 10.4: Viết chương trình tạo Menu cho phép chọn và thực hiện các chức năng bằng cách di chuyển mũi tên trên các hộp sáng, các thủ tục thực hiện xong quay trở lại Menu chính. Nhấn ESC để thoát khỏi chương trình.

```
USES  crt,graph;
Const   mau1 =15;
        mau2 =8;
        maumn=7;
        XTop=200;
        YTop=100;
        Dy=32;
        Dx=250;
Type MANG_MENU=Array[1..20] of string;{dung luu cac dong menu }
    MANG_THUTUC=Array[1..20] of Procedure;{dung luu cac thu tuc}
var  DongMN:MANG_MENU;
    ThuTuc:MANG_THUTUC;
    SoDong:byte;
Procedure Wait;
Var ch:Char;
```

```
BEGIN
   ch:=ReadKey;
END;
{$F+}
Procedure Modun1;
BEGIN
   Line(50,50,200,300);
   Wait;
END;
Procedure Modun2;
BEGIN
   Circle(200,200,100);
   Wait;
END;
Procedure Modun3;
Begin
   Ellipse(200,300,0,360,100,150);
   Wait;
End;
Procedure Modun4;
BEGIN
   Rectangle(50,50,200,300);
   Wait;
END;
Procedure Modun5;
BEGIN
   OutTextXY(50,50,'Chao mung cac ban den voi chuong trinh do hoa');
   Wait;
END;
Procedure Modun6;
BEGIN
   OutTextXY(50,50,'Day la Menu do hoa');
   Wait;
END;
Procedure Thoat;
BEGIN
 Halt;
END;
{$F-}
Procedure ThietLapDoHoa;
 var gd,gm:integer;
 Begin
   Gd:=0;
   InitGraph(gd,gm,'C:\BP\BGI');
 End;
Procedure Box(x1,y1,x2,y2:integer; MauVienTren,MauVienduoi,MauNen:byte);
{Ve nut menu}
  Var i:Byte;
 begin
   setfillstyle(1,MauNen);
```

```
   bar(x1,y1,x2,y2);
   setcolor(MauVienTren);
   For i:=0 to 1 do
     Begin
       moveto(x1-i,y2+i);
       lineto(x1-i,y1-i);
       lineto(x2+i,y1-i);
     End;
  setcolor(MauVienDuoi);
  For i:=0 to 1 do
    Begin
      moveto(x2+i,y1-i);
      lineto(x2+i,y2+i);
      lineto(x1-i,y2+i);
    End;
 end;
Procedure Ve_menu(Xdau,Ydau,DeltaX,DeltaY:Word;
                  chon,SoDong:Byte;DongMN:MANG_MENU);
 Var i:Byte;
 Begin
   for i:=1 to SoDong do
     begin
       if i=chon then
            Box(Xdau,Ydau+i*DeltaY+6,Xdau+DeltaX,YDau+i*DeltaY+DeltaY,
                  mau2,mau1,maumn)
       Else
            Box(Xdau,Ydau+i*DeltaY+6,Xdau+DeltaX,YDau+i*DeltaY+DeltaY,
                  mau1,mau2,maumn);
       OutTextxy(Xdau+20,Ydau+15+i*DeltaY,DongMN[i]);
     end;
 End;

Procedure PullDown(x,y,DeltaX,DeltaY:Word;SoDong:Byte;
                   DongMenu:MANG_MENU;ThuTuc:MANG_THUTUC);
 Var sott,LuuSott,Chon,i:Byte;
     OK:Boolean;

Function Select(Xdau,Ydau,DeltaX,DeltaY:Word;SoDong:Byte):Byte;
  var ch:char; j:Byte;
  Begin
   While True do
    Begin
       If KeyPressed then
       Begin
       ch:=readkey;
       case ch of
          #13: Begin {ENTER}
                 select:=Sott;
                 Exit;
               End;
```

```
          #72:Begin
               LuuSott:=Sott;
               Sott:=Sott-1;
               if Sott<1 then Sott:=SoDong;
               Select:=Sott;
              Box(XTop,YTop+LuuSoTT*DeltaY+6,
                    Xdau+DeltaX,Ydau+LuuSoTT*DeltaY+DeltaY,
                    Mau1,Mau2,maumn);
               Outtextxy(Xdau+20,Ydau+15+LuuSoTT*DeltaY,
                         DongMN[LuuSoTT]);
               Box(Xdau,Ydau+SoTT*DeltaY+6,
                    Xdau+DeltaX,Ydau+SoTT*DeltaY+DeltaY,
                    Mau2,Mau1,maumn);
               Outtextxy(Xdau+20,Ydau+15+SoTT*DeltaY,
                         DongMN[SoTT]);
             End;
          #80:
             Begin
               LuuSott:=Sott;
               Sott:=Sott+1;
               if Sott>SoDong then Sott:=1;
               Select:=Sott;
               Box(Xdau,Ydau+LuuSoTT*DeltaY+6,
                    Xdau+DeltaX,Ydau+LuuSoTT*DeltaY+DeltaY,
                    Mau1,Mau2,maumn);
               Outtextxy(Xdau+20,Ydau+15+LuuSoTT*DeltaY,
                          DongMN[LuuSoTT]);
               Box(Xdau,Ydau+SoTT*DeltaY+6,
                    Xdau+DeltaX,Ydau+SoTT*DeltaY+DeltaY,
                    Mau2,Mau1,maumn);
               Outtextxy(Xdau+20,Ydau+15+SoTT*DeltaY,
                          DongMN[SoTT]);
             End;
          #27: {ESC}
             Begin
               OK:=False; Exit;
             End;
        end; { of case key }
       End;
     End;
  End;
Begin {PullDown}
   Sott:=1;   OK:=TRUE;
   Ve_menu(X,Y,DeltaX,DeltaY,Sott,SoDong,DongMenu);
   While OK do  { lap khong dieu kien }
     Begin
        Chon:=select(x,y,DeltaX,DeltaY,SoDong);
        For i:=1 to SoDong do
          If (i=Chon)and OK Then
            Begin
```

```
            ClearDevice;
            ThuTuc[i];
            ClearDevice;
              Ve_Menu(X,Y,DeltaX,DeltaY,Sott,SoDong,DongMenu);
           End;
    end;{ of While }
 End;
BEGIN
  SoDong:=7;
  DongMN[1]:='VE DOAN THANG ';
  DongMN[2]:='VE DUONG TRON';
  DongMN[3]:='VE ELLIPSE';
  DongMN[4]:='VE HINH CHU NHAT';
  DongMN[5]:='VIET LOI CHAO';
  DongMN[6]:='VIET DONG QUANG CAO';
  DongMN[7]:='THOAT KHOI CHUONG TRINH';
  ThuTuc[1]:=Modun1;
  ThuTuc[2]:=Modun2;
  ThuTuc[3]:=Modun3;
  ThuTuc[4]:=Modun4;
  ThuTuc[5]:=Modun5;
  ThuTuc[6]:=Modun6;
  ThuTuc[7]:=Thoat;
  ThietLapDoHoa;
  SetBKcolor(LightBlue);
  PullDown(XTop,YTop,DX,DY,SoDong,DongMN,ThuTuc);
 CloseGraph;
END.
```

Bài tập 10.5: Vẽ hai hình

Sau đó, viết chương trình thực hiện chuyển động của miệng cá.

```
Uses Crt,Graph;
Type ProType=Procedure;
Var Gd,Gm:integer;
    page1,page2:word;
    Hinh:Array[0..1] of ProType;
    Xc,Yc,r:Integer;
    i:Byte;
{$F+}
Procedure HinhCa1;
Begin
  SetColor(15);
  PieSlice(Xc,Yc,30,330,R); {Ve bung ca}
  SetColor(0);
  Circle(Xc + R div 2,Yc - R div 2,4); {Mat ca}
 End;
```

```
Procedure HinhCa2;
 Begin
   SetColor(15);
   PieSlice(Xc,Yc,15,345,R); {Ve bung ca}
   SetColor(0);
   Circle(Xc + R div 2 ,Yc - R div 2,4); {Mat ca}
 End;
{$F-}
Begin
  gd:=4;
  InitGraph(gd,gm,'');
  Xc:=GetMaxX div 2;
  Yc:=GetMaxY div 2;
  R:=50; i:=0;
  Hinh[0]:=HinhCa1;
  Hinh[1]:=HinhCa2;
  page1:=0; page2:=1;
  Repeat
    SetVisualPage(page1);
    SetActivePage(page2);
    i:=1-i;
    Hinh[i]; Delay(200);
    page1:=1-page1;
    page2:=1-page2;
  Until  KeyPressed;
  CloseGraph;
End.
```

Bài tập 10.6: Viết chương trình tạo một dòng chữ chạy ngang qua màn hình.

```
Uses crt,graph;
Var gd,gm:integer;
Procedure Run(s:string);
  var page:byte;x,y:integer;
  Begin
    page:=1;
    x:=getmaxx;y:=getmaxy div 3;
    Settextjustify(0,1);
    Setwritemode(xorput);
    Setactivepage(page);
    Repeat
      Outtextxy(x,y,s);
      Setvisualpage(page);
      page:=not page;
      setactivepage(page);
      delay(10);
      Outtextxy(x+1,y,s);
      x:=x-1;
      if x<-textwidth(s) then x:=getmaxx;
    Until (keypressed) and (readkey=#27);
  end;
```

```
Begin
     gd:=4;
     Initgraph(gd,gm,'C:\BP\bgi');
     setcolor(14);
     settextstyle(1,0,5);
     Run('Pham Anh Phuong');
     Closegraph;
End.
```

Bài tập 10.7: Viết chương trình vẽ mô hình chiếc đĩa bay chuyển động ngẫu nhiên trên màn hình.

```
Uses crt; Graph;
Const r = 20; StartX = 100; StartY = 50;
Procedure ThietLapDohoa;
Var Gd,Gm:Integer;
Begin
   Gd:=0;
   InitGraph(Gd,Gm,'D:\BP\BGI');
End;
Procedure Move(Var x,y:Integer);
   Var Step:Integer;
   Begin
      Step:=Random(2*r);
      If Odd(Step) Then Step:=-Step;
      x:=x+Step;
      Step:=Random(r);
      If Odd(Step) Then Step:=-Step;
      y:=y+Step;
   End;
Procedure VeDiaBay;
   Begin
      Ellipse(StartX,StartY,0,360,r,(r div 3)+2);
      Ellipse(StartX,StartY-4,190,357,r,r div 3);
      Line(StartX+7,StartY-6,StartX+10,StartY-12);
      Line(StartX-7,StartY-6,StartX-10,StartY-12);
      Circle(StartX+10,StartY-12,2);
      Circle(StartX-10,StartY-12,2);
   End;
Procedure Play;
   Var x1,y1,x2,y2,size:Word;
        x,y:Integer;
        P:Pointer;
   Begin
      VeDiaBay;
      x1:=StartX - (r+1);
      y1:=StartY - 14;
      x2:=StartX + r + 1;
      y2:=StartY + (r div 3) + 3;
      (* Lưu và xóa ảnh *)
      size:=ImageSise(x1,y1,x2,y2);
      GetMem(p,size);
      GetImage(x1,y1,x2,y2,P^);
```

```
      PutImage(x,y,P^,XORPut); { Xóa ảnh }
      x:=GetMaxX div 2;
      y:=GetMaxY div 2;
      (* Di chuyển đĩa bay *)
      Repeat
         PutImage(x,y,P^,XORPut); { Vẽ đĩa bay }
         Delay(200);
         PutImage(x,y,P^,XORPut); { Xóa đĩa bay }
         Move(x,y);
      Until KeyPressed;
      FreeMem(p,size);  { Giải phóng vùng nhớ }
   End;
BEGIN
   ThietLapDoHoa;
   Play;
   CloseGraph;
END.
```

Bài tập 10.8: Viết chương trình để vẽ đa giác đều có n đỉnh.

Ý tưởng:

Khi vẽ một đa giác đều N đỉnh, các đỉnh này nằm trên một đường tròn (O,R) đồng thời khoảng cách giữa hai đỉnh và tâm tạo thành một góc nhọn không đổi có giá trị là 2*Pi/N.

Giả sử đỉnh thứ nhất của đa giác nằm trên đường thẳng tạo với tâm một góc 00, đỉnh thứ hai tạo một góc 2*Pi/N và đỉnh thứ i sẽ tạo một góc là 2*Pi(i-1)/N.

Một cách tổng quát, ta tạo một mảng để chứa tọa độ các đỉnh.

Const Max = <Giá trị>;
Type Mang = ARRAY[1..Max] of PointType;
Var P:Mang;

Giả sử chọn P0: PointType là tọa độ tâm của đa giác thì đỉnh thứ i của đa giác sẽ tạo một góc là:

Angle:=2*Pi*(i-1)/N

Nhưng nếu đa giác này có đỉnh đầu tiên tạo một góc bằng A0 thì:

Angle:=2*Pi*((i-1)/N + A0/360)

Và tọa độ các đỉnh này trên màn hình là:

P[i].x := P0.x + R*cos(Angle)
P[i].y := P0.y - R*sin(Angle)

Ta xây dựng thủ tục để tự động lưu các đỉnh của đa giác đều vào mảng P. Trong đó: P0 là tọa độ tâm, A0 là góc bắt đầu, R là bán kính, N là số đỉnh (3<N<Max).

```
Uses Crt,Graph;
Const Max = 10;
Type Mang = Array[1..Max] of PointType;
Var   A0,R:real;
      N:Byte;
      P0:PointType;   P:Mang;
Procedure ThietLapDohoa;
Var Gd,Gm:Integer;
Begin
   Gd:=0;
   InitGraph(Gd,Gm,'D:\BP\BGI');
End;
Procedure TaoDinh(R,A0:real;N:Byte;P0:PointType;Var P:MANG);
 var  i:Byte;
```

```
        Angle:real;
 Begin
   If (n<3)or(n>=Max) then
    Begin
      Writeln('Khong tao duoc tap dinh!');
      Exit;
    End;
   For i:=1 to n do
     With P[i] do
      Begin
        Angle:=2*Pi*((i-1)/n + A0/360);
        x:=P0.x + Round(R*Cos(Angle));
        y:=P0.y - Round(R*Sin(Angle));
      End;
   P[n+1]:=p[1];
 End;
BEGIN
    Write('Nhap so dinh cua da giac deu: n= '); Readln(N);
    ThietLapDoHoa;
    P0.x:=GetMaxX div 2;
    P0.y:=GetMaxY div 2;
    A0:=90;
    R:=GetMaxY div 4;
    TaoDinh(R,A0,5,P0,P);
    DrawPoly(5,P);
    CloseGraph;
END.
```

Bài tập 10.9: Viết chương trình vẽ đồ thị hàm số sau: $f(x) = ax^2 + bx + c$.

Ý tưởng:

Bước 1: Xác định đoạn cần vẽ [Min,Max].

Bước 2: Đặt gốc tọa độ lên màn hình (x0,y0).
Chia tỉ lệ vẽ trên màn hình theo hệ sô k.
Chọn số gia dx trên đoạn cần vẽ.

Bước 3: Chọn điểm xuất phát: x = Min, tính f(x).
Đổi qua tọa độ màn hình và làm tròn:
x1:=x0 + Round(x.k);
y1:=y0 - Round(y.k);
Di chuyển đến (x1,y1): MOVETO(x1,y1);

Bước 4: Tăng x lên: x:=x + dx;
Đổi qua tọa độ màn hình và làm tròn:
x2:=x0 + Round(x.k);
y2:=y0 - Round(y.k);
Vẽ đến (x2,y2): LINETO(x2,y2);

Bước 5: Lặp lại bước 4 cho đến khi x > Max thì dừng.

```
Uses Crt,Graph;
 var a,b,c,Max,Min:real;
Procedure ThietLapDohoa;
Var Gd,Gm:Integer;
Begin
   Gd:=0;
```

```
   InitGraph(Gd,Gm,'D:\BP\BGI');
End;
Function F(x:real):real;
 Begin
   F:=a*x*x + b*x + c;
 End;
Procedure VeDoThi(Min,Max:real);
 var x1,y1:integer;
     dx,x,k:real;
     x0,y0:word;
 Begin
   x0:=GetMaxX div 2;
   y0:=GetMaxY div 2;
   K:=GetMaxX/30;
   dx:=0.001;
   x:=Min;
   x1:=x0 + Round(x*k);
   y1:=y0 - Round(F(x)*k);
   Moveto(x1,y1);
   While x<Max do
     Begin
       x:=x+dx;
       x1:=x0 + Round(x*k);
       y1:=y0 - Round(F(x)*k);
       LineTo(x1,y1);
     End;
 End;
BEGIN
 Write('Nhap a= '); Readln(a);
 Write('Nhap b= '); Readln(b);
 Write('Nhap c= '); Readln(c);
 ThietLapDoHoa;
 Min:=-10; Max:=10;
 {Vẽ trục tọa độ}
 Line(GetMaxX Div 2,1,GetMaxX Div 2,GetMaxY);
 Line(1,GetMaxY Div 2,GetMaxX,GetMaxY Div 2);
 VeDoThi(Min,Max);
 Repeat Until KeyPressed;
 CloseGraph;
END.
```

Bài tập 10.10: Vẽ hình bông hoa.

Ý tưởng:

Dùng tọa độ cực. Giả sử ta có tọa độ cực trong đó:

Trục cực: Ox

Góc quay: [redacted]

thì tọa độ cực của một điểm trong mặt phẳng là cặp (x,y) với:

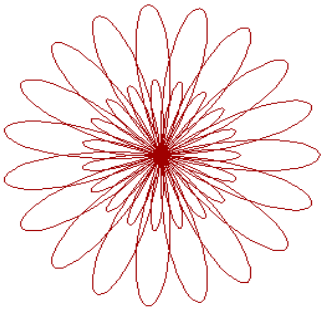
x = f([redacted]Cos([redacted]

y = f([redacted]Sin([redacted]

Trong đó: f([redacted]là phương trình do ta qui định.

Ví dụ:

f(= k.Cos(n: Hình bông hoa.



Hình bông hoa

f(= a.a>0) : Đường xoắn ốc Acsimet.

f(= k.(1+Cos(: Hình trái tim.

```
Uses Crt,Graph;
 var R,chuky:real;
Procedure ThietLapDohoa;
Var Gd,Gm:Integer;
Begin
    Gd:=0;
    InitGraph(Gd,Gm,'D:\BP\BGI');
End;
Function F(R,Alpha:real):real; { Tính hàm f()}
 Begin
   F:=R*cos(19*Alpha/3)+5;
 End;
Procedure VeHinh(ChuKy:real);
 var x1,x2,y1,y2:integer;
     a,Alpha,k:real;
     x0,y0:word;
 Begin
   x0:=GetMaxX div 2;
   y0:=GetMaxY div 2;
   K:=GetMaxX/50;
   a:=Pi/180;
   Alpha:=0;
   x1:=x0 + Round(F(R,Alpha)*k*cos(Alpha));
   y1:=y0 - Round(F(R,Alpha)*k*sin(Alpha));
   Moveto(x1,y1);
   While Alpha<ChuKy do
     Begin
       Alpha:=Alpha+a;
       x1:=x0 + Round(F(R,Alpha)*k*cos(Alpha));
       y1:=y0 - Round(F(R,Alpha)*k*sin(Alpha));
       LineTo(x1,y1);
       Delay(10);
     End;
 End;
BEGIN
 ThietLapDoHoa;
 R:=15; chuky:=4*Pi;
```

```
 VeHinh(chuky);
 repeat until KeyPressed;
 CloseGraph;
END.
```

Bài tập 10.11: Viết chương trình vẽ cung Koch. Các bước phát sinh của cung Koch được thực hiện trong hình sau:

- Bắt đầu từ đường ngang K_0 có độ dài bằng 1.
- Để tạo cung bậc-1(gọi là K_1), chia đường thành ba phần và thay đoạn giữa bằng tam giác đều có cạnh dài 1/3. Bây giờ, toàn bộ đường cong có độ dài 4/3.
- Cung bậc-2 K_2 có được bằng cánh dựng tiếp các tam giác đều từ 4 đoạn của K1. Vì mỗi đoạn có độ dài tăng 4/3 lần nên toàn bộ cung dài ra 4/3 lần.

(a) K_0

(b) K_1

(c) K_2

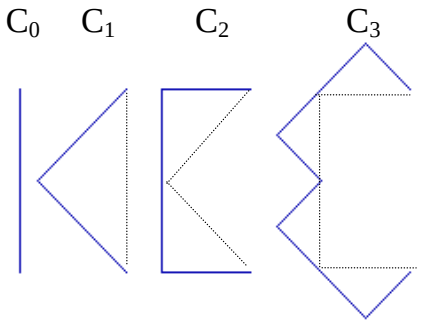
Ý tưởng:

Từ hình (b) ta thấy rằng, đầu tiên hướng vẽ quay trái 60^0, rồi quay phải 120^0, cuối cùng quay trái 60^0 để trở về hướng ban đầu.

```
Uses  Crt,Graph;
Var n:Integer;
   Goc,length:real;
Procedure ThietLapDohoa;
Var gd,gm:integer;
Begin
  gd:=0;
  InitGraph(gd,gm,'D:\bp\bgi');
End;
Procedure   Koch(dir,len:real;n:integer);
const  rads=0.017453293;
Begin
  If n>0 Then
    Begin
      Koch(dir,len/3,n-1);
      dir:=dir+60; {Quay phải 60 độ}
      Koch(dir,len/3,n-1);
      dir:=dir-120; {Quay trái 120 độ}
      Koch(dir,len/3,n-1);
      dir:=dir+60; {Quay phải 60 độ}
      Koch(dir,len/3,n-1);
    End
  else LineRel(Round(len*cos(rads*dir)),Round(len*sin(rads*dir)));
end;
Begin
  ThietLapDoHoa;
  n:=4;
  Goc:=180;
  Length:=150;
  Moveto(300,200);
  Koch(Goc,Length,n);
  Repeat until  keypressed;
```

```
  Closegraph;
END.
```

Bài tập 10.12: Viết chương trình tạo ra C-cung dựa trên sự tinh chế tương tự của một đoạn thẳng theo hình sau:



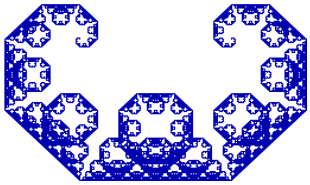
Ý tưởng:

Để có dạng phát sinh kế tiếp, mỗi đoạn thẳng được thay bởi một "hình gãy" gồm 2 đoạn ngắn hơn tạo với nhau một góc 90^0. Các đoạn mới có độ dài bằng $1/\sqrt{2}$ lần đoạn ở bước trước.

Xét hướng vẽ ở một đầu của đoạn thẳng. Để vẽ hình gãy, hướng vẽ quay trái 45^0, vẽ một đoạn, quay phải 90^0, vẽ đoạn thứ hai và sau đó trở về hướng cũ bằng cách quay góc 45^0.

```
Uses  graph,crt;
Procedure ThietLapDohoa;
Var    gd,gm,gr:integer;
Begin
  gd:=0;
  Initgraph(gd,gm,'D:\bp\bgi');
End;
PROCEDURE     VeC_Cung;
Var n:Integer;
      Goc,length:real;
    Procedure   Rong(dir,len:real;n:integer);
     const  d=0.7071067;
            rads=0.017453293;
     begin
       if n>1 then
         begin
           dir:=dir+45;
           Rong(dir,len*d,n-1);
           dir:=dir-90;
           Rong(dir,len*d,n-1);
           dir:=dir+45;
         end
       else LineRel(Round(len*cos(rads*dir)),Round(len*sin(rads*dir)));
     end;
Begin
    n:=15;
    Goc:=0;
    Length:=130;
    Moveto(200,200);
    Rong(Goc,Length,n);
    repeat until  keypressed;
```

```
  End;
BEGIN
  ThietLapDoHoa;
  VeC_Cung;
  Closegraph;
END.
```



C Cung

Bài tập 10.13: Viết chương trình vẽ tập Mandelbrot - là một hình trong mặt phẳng phức. Tập Mandelbrot được phát sinh theo công thức sau:

z [redacted] $+ c$ (*)

Tập hợp Mandelbrot là tập bao gồm những số phức c sao cho z^2+c vẫn hữu hạn với mọi lần lặp.

Ý tưởng:

Ta chọn số phức cố định c và tính biểu thức z2+c với z là số phức biến đổi.

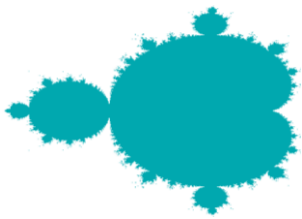
Nếu chọn z = 0 thì $z^2+c = c$. Thay z vào công thức (*) ta được c^2+c.

Tiếp tục thay z bằng giá trị mới, ta lại có: (c2+c)2+c, ...

Cứ như vậy, ta thu được một dãy vô hạn các số z.

```
Uses crt,graph;
Const row=1;
         col=1;
Var   x1,y1,x2,y2,kx,ky:real;
        Gioihan:Byte;
        x0,y0:word;
    Diemduoi,Diemtren:Integer;
Procedure ThietLapDohoa;
Var    gd,gm,gr:integer;
Begin
  gd:=0;
  Initgraph(gd,gm,'D:\bp\bgi');
End;
Procedure KhoiTao;
 Begin
   Diemduoi:=GetMaxX;
   Diemtren:=GetMaxY;
   x1:=-2; y1:=-1.25;
   x2:=0.5; y2:=1.25;
   kx:=(x2-x1)/diemduoi;
   ky:=(y2-y1)/diemtren;
   Gioihan:=50;
 End;
Procedure ManDelbrot;
 var  dong,cot,dem:integer;
      P0,Q0,Modun,x,y,Aux:real;
 Begin
  cot:=0;
  While cot<=diemduoi do
```

```
  Begin
    P0:=x1+cot*kx;
    dong:=0;
    While dong<=(diemtren div 2) do
      Begin
       Q0:=y1+dong*ky;
       x:=0; y:=0;
       dem:=1; Modun:=1;
       While (dem<=gioihan)and(modun<4) do
         Begin
           Aux:=x;
           x:=x*x-y*y +P0;
           y:=2*y*Aux + Q0;
           Modun:=x*x + y*y;
           dem:=dem+1;
         End;
       If Modun<4 Then
         Begin
          PutPixel(cot,dong,3);
          PutPixel(cot,diemtren-dong,3);
         End;
       dong:=dong+row;
      End;
     cot:=cot+col;
  End;
 End;
Begin
 ThietLapDohoa;
 KhoiTao;
 Mandelbrot;
 readln;
 CloseGraph;
End.
```



Tập MandelBrot

Bài tập 10.14: Viết chương trình mô phỏng phép quay một tam giác quanh gốc tọa độ.

Ý tưởng:

Ma trận của phép quay quanh gốc tọa độ: R = $\begin{bmatrix} Cos(\alpha) & Sin(\alpha) \\ Sin(\alpha) & Cos(\alpha) \end{bmatrix}$

$\Leftrightarrow \begin{cases} x' = x.Cos(\alpha) - y.Sin(\alpha) \\ y' = x.Sin(\alpha) + y.Cos(\alpha) \end{cases}$

```
Uses crt,Graph;
 Type ToaDo=Record
       x,y:real;
```

```
     End;
 var k,Alpha,goc:real;
     P,PP,PPP,P1,P2,P3:ToaDo;
     x0,y0:word;
     ch:char;
Procedure ThietLapDohoa;
Var    gd,gm,gr:integer;
Begin
  gd:=0;
  Initgraph(gd,gm,'D:\bp\bgi');
End;

Procedure VeTruc;
  Begin
     Line(GetMaxX div 2,0,GetMaxX div 2,GetMaxY);
     Line(0,GetMaxY div 2,GetMaxX,GetMaxY div 2);
  End;
Procedure VeHinh(P1,P2,P3:ToaDo);
  Begin
    Line(x0+Round(P1.x*k),y0-Round(P1.y*k),
           x0+Round(P2.x*k),y0-   Round(P2.y*k));
    Line(x0+Round(P2.x*k),y0-Round(P2.y*k),
           x0+Round(P3.x*k),y0-   Round(P3.y*k));
    Line(x0+Round(P3.x*k),y0-Round(P3.y*k),
           x0+Round(P1.x*k),y0-   Round(P1.y*k));
  End;
Procedure QuayDiem(P:ToaDo;Alpha:real; var PMoi:ToaDo);
  Begin
    PMoi.x:=P.x*cos(Alpha)-P.y*sin(Alpha);
    PMoi.y:=P.x*sin(Alpha)+P.y*cos(Alpha);
  End;
Procedure QuayHinh(P1,P2,P3:ToaDo;Alpha:real;
                           var P1Moi,P2Moi,P3Moi:ToaDo);
  Begin
   QuayDiem(P1,Alpha,P1Moi);
   QuayDiem(P2,Alpha,P2Moi);
   QuayDiem(P3,Alpha,P3Moi);
  End;
BEGIN
  ThietLapDoHoa;
  x0:=GetMaxX div 2;
  y0:=GetMaxY div 2;
  k:=GetMaxX/50;
  Vetruc;
  P.x:=5; P.y:=3; PP.x:=2; PP.y:=6; PPP.x:=6; PPP.y:=-4;
  P1:=P; P2:=PP; P3:=PPP;
  Alpha:=0; goc:=Pi/180;
  SetWriteMode(XORPut);
  VeHinh(P,PP,PPP);
  Repeat
```

```
    ch:=readkey;
    if ord(ch)=0 then ch:=readkey;
    case Upcase(ch) of
      'K': Begin
             VeHinh(P1,P2,P3);
             Alpha:=Alpha-goc;
             QuayHinh(P,PP,PPP,Alpha,P1,P2,P3);
             VeHinh(P1,P2,P3);
           End;
      'M': Begin
             VeHinh(P1,P2,P3);
             Alpha:=Alpha+goc;
             QuayHinh(P,PP,PPP,Alpha,P1,P2,P3);
             VeHinh(P1,P2,P3);
           End;
    End;
  Until ch=#27;
  CloseGraph;
END.
```

BÀI TẬP TỰ GIẢI

Bài tập 10.15: Viết chương trình vẽ bàn cờ quốc tế lên màn hình.

Bài tập 10.16: Viết chương trình vẽ một chiếc xe ô tô (theo hình dung của bạn) và cho nó chạy ngang qua màn hình.

Gợi ý:

Dùng kỹ thuật lật trong màn hình hoặc di chuyển vùng màn hình.

Bài tập 10.17: Viết chương trình vẽ lá cờ tổ quốc đang tung bay.

Gợi ý:

Dùng kỹ thuật lật trong màn hình.

Bài tập 10.18: Viết chương trình nhập vào n học sinh của một lớp học bao gồm 2 trường sau: Họ tên, điểm trung bình.

a/ Hãy thống kê số lượng học sinh giỏi, khá, trung bình và yếu.

b/ Vẽ biểu đồ thống kê số lượng học sinh giỏi, khá, trung bình và yếu theo 2 dạng: biểu đồ cột (column) và biểu đồ bánh tròn (Pie).

Bài tập 10.19: Viết chương trình để vẽ đồ thị của các hàm số sau:

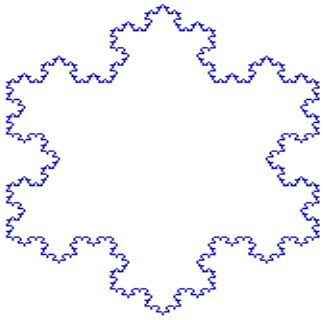
a/ $y = ax^3 + bx^2 + cx + d$

b/ $y = ax^4 + bx^3 + cx^2 + dx + e$

c/ $y = \frac{ax + b}{cx + d}$

d/ $y = \frac{ax^2 + bx + c}{dx + e}$

Bài tập 10.20: Hình vẽ cung Koch dựa trên 3 cạnh của tam giác đều như hình sau:



Bài tập 10.21: Viết chương trình để vẽ đường xoắn ốc.

Gợi ý:

Dùng tọa độ cực.

Bài tập 10.22: Viết chương trình vẽ cái đồng hồ đang hoạt động.

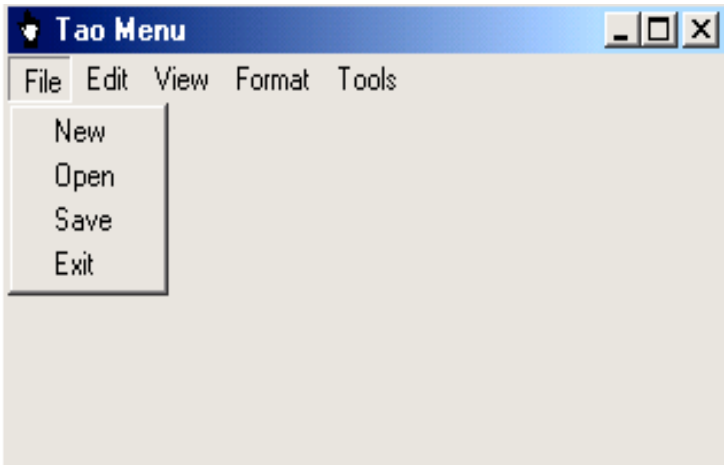
Bài tập 10.23: Viết chương trình mô phỏng chuyển động của trái đất xung quanh mặt trời và đồng thời chuyển động của mặt trăng xung quanh trái đất.

Gợi ý:

Dùng ma trận của phép quay.

Bài tập 10.24: Xây dựng một thư viện (Unit) chứa tất cả các bài tập trong chương này.

Bài tập 10.25: Viết chương trình tạo Menu đồ họa giống như các Menu trong môi trường WINDOWS (xem hình).



MỤC LỤC

Lời mở đầu....1
Chương 1: CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ LẬP TRÌNH PASCAL 2
Chương 2: CÁC KIỂU DỮ LIỆU CƠ BẢN – KHAI BÁO HẰNG, BIẾN, KIỂU, BIỂU THỨC VÀ CÂU LỆNH
I. Các kiểu dữ liệu cơ bản....6
II. Khai báo hằng....8
III. Khai báo biến 8
IV. Định nghĩa kiểu 9
V. Biểu thức 9
VI. Câu lệnh....9
Bài tập mẫu....11
Bài tập tự giải 12
Chương 3: CÁC CÂU LỆNH CÓ CẤU TRÚC
I. Lệnh rẽ nhánh....15
II. Lệnh lặp 16
Bài tập mẫu....17
Bài tập tự giải 24
Chương 4: CHƯƠNG TRÌNH CON: THỦ TỤC VÀ HÀM
I. Khái niệm về chương trình con 27
II. Cấu trúc chung của một chương trình có sử dụng CTC....27
III. Biến toàn cục và biến địa phương 28
IV. Đệ qui....29
V. Tạo thư viện (UNIT)....31
Bài tập mẫu....33
Bài tập tự giải 36
Chương 5: DỮ LIỆU KIỂU MẢNG
I. Khai báo mảng....38
II. Xuất nhập trên dữ liệu kiểu mảng 38
Bài tập mẫu....38
Bài tập tự giải 50
Chương 6: XÂU KÝ TỰ
I. Khai báo kiểu xâu ký tự....53
II. Truy xuất dữ liệu kiểu String....53
III. Các phép toán trên xâu ký tự 53
IV. Các thủ tục và hàm về xâu ký tự 53
Bài tập mẫu....54
Bài tập tự giải 60

Chương 7: KIỂU BẢN GHI
I. Khai báo dữ liệu kiểu bản ghi....63
II. Xuất nhập dữ liệu kiểu bản ghi....63
Bài tập mẫu....63
Bài tập tự giải 68
Chương 8: KIỂU FILE
I. Khai báo 70
II. Các thủ tục và hàm chuẩn....70
III. File văn bản 72

IV. File không định kiểu....73
Bài tập mẫu....74
Bài tập tự giải....85
Chương 9: KIỂU CON TRỎ
I. Khai báo....91
II. Làm việc với biến động....91
III. Danh sách động....92
Bài tập mẫu....94
Bài tập tự giải....108
Chương 10: ĐỒ HỌA
I. Màn hình trong chế độ đồ hoạ....113
II. Khởi tạo và thoát khỏi chế độ đồ hoạ....113
III. Toạ độ và con trỏ trên màn hình đồ hoạ....115
IV. Đặt màu trên màn hình đồ hoạ....115
V. Cửa sổ trong chế độ đồ hoạ....115
VI. Viết chữ trong chế độ đồ họa....116
VII. Vẽ các hình cơ bản....116
VIII. Tô màu các hình....117
IX. Các kỹ thuật tạo hình chuyển động....119
Bài tập mẫu....120
Bài tập tự giải....141
Mục lục....143