Lời nói đầu!

Ngôn ngữ Pascal do giáo sư N. Wirth đưa ra năm 1970, là một ngôn ngữ có cấu trúc, thường dùng để giảng dạy cho Sinh viên và Học sinh khi mới bước vào lĩnh vực học lập trình.

Chúng ta học Pascal bởi 3 lý do sau:

- Ngôn ngữ Pascal hiện ở Việt nam vẫn đang phổ biến
- Tạo cho người học một tác phong "kỷ luật lập trình" cấu trúc
- Làm quen với một số dữ liệu, đặc biệt là dữ liệu có cấu trúc giúp cho người học tiếp tục tìm hiểu tiếp C ++, Visual Basic và Java ở những giáo trình sau.

Đây là phần cơ bản nhất, giáo trình không được vào phần ***cấp phát động*** và ***hướng đối tượng*** bởi vì: thứ nhất số trình về học phần này không nhiều; thứ 2 là trong C ++ và Java sẽ được dạy ở những học kỳ sau có chứa đựng 2 nội dung đó; thứ ba là vì tác giả muốn nhấn mạnh tư tưởng “Lập trình cấu trúc” nên không muốn làm “loãng” ra.

Giáo trình này chạy trên các phiên bản của Turbo Pascal 5.0, 5.5, 6.0 và 7.0. Tất nhiên cũng cần phải nói thêm là Turbo Pascal 7.0 có gây lỗi trên những bộ vi xử lý Pentium II, III, v.v.. có tốc độ lớn, bởi vì ông chủ của nó (Borland) đã không cập nhật nữa vì ý muốn chuyển sang dòng ngôn ngữ khác. Tuy vậy bạn đừng lo, bạn có thể tự sửa các lỗi ấy bằng những thao tác đơn giản, hoặc né tránh không dùng những lệnh gây lỗi đó.

Các bạn nên tập trung bước đầu làm quen với "lập trình cấu trúc" này, nó sẽ là cầu nối tốt giúp bạn về sau khi học lập trình bằng những ngôn ngữ khác trong môi trường Windows.

Khi thực hành trên máy bạn còn được cung cấp quyển bài tập Pascal mà trong đó các bài tập cùng theo trật tự của phần lý thuyết này.

Chúc các bạn thành đạt!

Chương 1. Các thành phần cơ bản của ngôn ngữ

1.1 Bộ ký tự, từ khoá, tên gọi

Các ngôn ngữ bất kỳ đều được xây dựng dựa trên một bộ các ký tự, từ các ký tự đó mà các từ có nghĩa được tạo thành. Tiếp theo là các qui tắc để tạo thành câu để diễn tả các hành vi, sự việc nghĩa là phải tuân thủ cú pháp (syntax) và ngữ pháp (grammar) của ngôn ngữ đó. ***Ngôn ngữ Pascal cũng theo quy cách đó.***

a. Bộ ký tự:

Bộ chữ cái la tinh: 26 chữ cái lớn A, B, C, D...Z.
26 chữ cái nhỏ a, b, c, d,..., z.
Ký tự gạch nối _
Các chữ số thập phân: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Các ký tự toán học: +, -, *, /, = ,<, > , (,)...
Các ký tự đặc biệt: ., ;, :, [,], ?, %, @,...
Dấu cách: (Space) dùng để ngăn cách các từ.

b. Một số từ của Pascal được gọi là từ khoá (keyword).

Các từ khoá này bao gồm:

- **Từ khoá chung**:
  Program, Begin, End, Procedure, Function.
- **Từ khoá để khai báo:**
  Conts, Var, Label, Type, Array, String, Record, File of
- **Từ khoá lệnh điều kiện, chọn và lặp:**
  if... then...else..., Case...of...
  For... to...do và For...Downto...do...
  While... do...
  Repeat... until...
- **Từ khoá điều khiển:**
  With, Goto
- **Từ khoá toán tử:**
  And, Or, Not, In, Div, Mod
  V.v..

Các từ khoá có thể viết bằng chữ to hoặc chữ nhỏ, hoặc pha lẫn ví dụ: Begin, BEGIN, begin đều như nhau.

d. Tên (định danh - identifier):

Tên hoặc định danh dùng để chỉ tên hằng, biến, kiểu, tên chương trình con...

Tên được tạo thành từ các chữ cái và các chữ số nhưng chữ đầu phải là chữ cái (có thể dùng thêm dấu gạch dưới). Nên đặt tên sao cho có ý nghĩa và dễ nhận biết. Ví dụ để đặt tên biến là diện tích hình tròn ta nên dùng: S hoặc DienTichHinhTron hoặc Dien _Tich_Hinh_tron v.v.. Các tên viết như sau là sai, Pascal không chấp nhận:

#DienTich Sai vì tên bắt đầu bằng ký tự không được phép
BEGIN Sai vì tên trùng với từ khoá
Dien Tich Sai vì có dấu cách (Space)
3XY Sai vì tên bắt đầu bằng chữ số,

Một số tên dùng cho tên hàm như EXP, SIN, COS,...) được gọi là các ***tên chuẩn***. Sự khác nhau giữa ***tên chuẩn*** và ***từ khoá*** là: người sử dụng có thể định nghĩa lại các ***tên chuẩn*** vào việc khác nếu cần. Còn ***từ khoá*** thì không được phép như vậy.

Một số tên chuẩn của Pascal:

Boolean, Char, Integer, Real, Byte, Text.

False, True, MaxInt.

Abs, artctan, Chr, Cos, Sin, Eof, Eoln, Exp, Ln, Odd, Ord, Round, Trunc, Sqr, Sqrt, Pred, Succ.

Dispose, New, Get, Put, Read, Realn, Write, Writeln, Reset, Rewrite.

1.2 Ngăn cách các lệnh

Dấu chấm phẩy ";" được dùng để ngăn cách các câu lệnh của Pascal.

1.3 Lời chú thích (Comment)

Lời chú thích có thể đưa vào bất kỳ vị trí nào trong chương trình mà không ảnh hưởng đến các phần khác. Lời chú thích là dành cho người đọc, máy sẽ bỏ qua khi gặp nó. Lời chú thích được đặt bên trong mở và đóng móc {và} hoặc ở giữa cụm dấu (*và v*).

Trong chương trình nên viết những lời chú thích vào những vị trí thích hợp giúp cho con người dễ dàng kiểm tra chương trình.

1.4 Cấu trúc chung của chương trình Pascal

Chương trình là một dãy các lệnh, chỉ dẫn cho máy thực hiện một nhiệm vụ nào đó.

Chương trình Pascal gồm 4 phần:

- Tiêu đề của chương trình
- Phần gọi các đơn vị (Unit) chương trình
- Phần khai báo: hằng, biến, kiểu và chương trình con
- Thân chương trình chứa các lệnh

Trước hết chúng ta xem một chương trình đơn giản tính tổng của hai số như sau:

```
Program TongHaiSo; (*Tiêu đề chương trình *)
Uses Crt; (*gọi đơn vị Crtg *)
     Var
          Tong,a,b: Real;          (*Khai báo ba biến thực)
      Begin
          ClrScr; (*Xoá màn hìnhX *)
           Write('Hãy vào số thứ nhất a = ');
```

```
        Readln(a);              (*Vào giá trị của a *)
        Write('Hãy vào số thứ hai b = ');
        Readln(b);              (*Vào giá trị của b *)
        Tong:= a+b;             (*Tính tổng hai số *)
         (*Hiển thị kết quả lên màn hình *)
        Writeln(' Tổng của hai số là:', Tong);
    End.
```

Như vậy cấu trúc tổng quát một chương trình đầy đủ gồm các phần chính như sau:

```
Program TenChuongTrinh ;     (*Dòng tiêu đề *)
Uses  CRT, Printer;          (*Lời gọi sử dụng các đơn vị chương trìnhL *)
(*Phần khai báo dữ liệu và chương trình con *)
Label...
Const...
Type...
Var...
Procedure...          {Có thể có nhiều Procedure và nhiều Function }
Function...
(*Thân chương trình chính *)
Begin
....
(*Các lệnh được viết ở đây *)
End.
```

Phần tiêu đề của chương trình:

Phần này bắt đầu bằng từ khoá ***Program***, tiếp theo là tên do người lập trình đặt ra theo một ý nghĩa của nội dung chương trình. Cuối phần tên là dấu chấm phẩy";".

Phần tiêu đề có thể không có cũng được.

- ***Phần khai báo:***

Phần này dùng để mô tả các đối tượng dữ liệu. Có thể có các khai báo sau:

```
Uses                  (*Khai báo sử dụng các UnitK *)
Label                 (*Khai báo nhãnK *)
...
Const                 (*Khai báo hằng *)
...
Type                  (*Mô tả  kiểu dữ liệu M *)
...
Var                   (*Khai báo biến K *)
...
Procedure...          (*Khai báo các thủ tụcK *)
Function...           (*Khai báo các hàmK *)
```

Các phần khai báo có thể có hoặc không, tuỳ theo thực tế của từng bài toán.

Phần thân chương trình:

Thân chương trình nằm giữa hai từ khoá **Begin** và **End**. Sau từ khoá **End** là dấu chấm "." để báo kết thúc chương trình. Không có thân thì bất thành chương trình, thân chứa các lệnh để xử lý các đối tượng dữ liệu đã được mô tả trong phần khai báo.

Ví dụ chương trình sau chỉ có thân:

```
Begin
   Writeln (' Chào bạn đã đến với Turbo Pascal 7.0');
End.
```

1.5 Các bước cơ bản khi soạn một chương trình.

Để soạn một chương trình bằng ngôn ngữ Pascal. Thông thường trải qua các bước sau:

Bước 1: Bước soạn thảo chương trình

Dùng một chương trình soạn thảo văn bản nào đó để soạn chương trình. Khi soạn xong ta lưu chương trình lên đĩa. Chương trình viết bằng ngôn ngữ Pascal được gọi là chương trình nguồn (source program). Tên các tệp (file) chứa chương trình nguồn Pascal có dạng chung *.Pas.

Bước 2: Bước dịch chương trình

Dùng chương trình dịch (compiler) Pascal để dịch chương trình nguồn sang mã máy

(chứa trong các tệp *. COM, *.EXE, *.OBJ). Giai đoạn này cho phép ta phát hiện ra các lỗi cú pháp khi viết chương trình. Nếu có lỗi, chương trình dịch sẽ báo lỗi. Ta phải quay về bước 1 để sửa và quay lại bước 2. Nếu không có lỗi cú pháp, ta sang bước 3.

Bước 3: Chạy chương trình

(Ta sẽ thực hiện 3 bước đó liên tục trong toàn bộ giáo trình này)

Chương 2: Các kiểu dữ liệu cơ sở

2.1 Kiểu Logic (Boolean)

Trong cuộc sống chúng ta thường hay gặp loại dữ liệu chỉ có hai giá trị: đúng (True) hoặc sai (False). Ví dụ khi ta viết 10>25 thì đây là một mệnh đề có giá trị **False**. Hoặc 1<4 là mệnh đề có giá trị **True.**

Một giá trị thuộc kiểu Boolean chỉ nhận một trong hai giá trị logic: TRUE (đúng) hoặc FALSE (sai). TRUE và FALSE là hai giá trị được định nghĩa sẵn.

Các phép toán sau cho kết quả cũng là kiểu Boolean:

Phép AND (phép"và")

Phép OR (phép "hoặc")

Phép NOT (phép "phủ định")

Phép XOR (phép "hoặc triệt tiêu")

- Phép AND chỉ cho kết quả là TRUE khi và chỉ khi cả hai toán hạng đều TRUE.
- Phép OR chỉ cho kết quả là FALSE khi và chỉ khi cả hai toán hạng đều là FALSE.
- Phép XOR cho kết quả là TRUE khi giá trị hai toán hạng khác nhau. Còn nếu giá trị hai toán hạng giống nhau, sẽ cho kết quả là FALSE.

Ví dụ:

(3<5) AND (15>7) cho kết quả TRUE

(3<5) AND (15<7) cho kết quả FALSE

(1>3) OR (4<7) cho kết quả TRUE

(1>3) OR (4>7) cho kết quả FALSE

(1>3) XOR (4<7) cho kết quả TRUE

(1>3) XOR (4>7) cho kết quả FALSE

NOT (1>9) cho kết quả TRUE

NOT (1<9) cho kết quả FALSE

2.2 kiểu số nguyên (Integer)

Một giá trị có kiểu nguyên là một phần tử của tập các số nguyên mà ta có thể xử lý được trên máy, tất nhiên nó chỉ là một tập nhỏ của các số nguyên chứ không phải tất cả mọi số nguyên đều có thể xử lý được trên máy. Kiểu số nguyên được định nghĩa sẵn với từ khoá INTEGER.

Thông thường nhất, các số nguyên được biểu diễn bằng hai byte (16 bít) nên phạm vi của nó là từ - 32768 ($=-2^{15}$) đến +32767($=2^{15}-1$).

Các số nguyên được viết ra có thể có dấu dương (+) phía trước hoặc dấu (-), hoặc không có dấu.

Thí dụ:

+120, -267, -5, 13, 163...

Maxint là tên giá trị cực đại cho phép của kiểu nguyên, nghĩa là **Maxint** = +32767.

1. Các phép tính số học đối với số nguyên:

- Phép cộng và phép trừ: +, -
- Phép nhân: *
- Phép chia lấy kết quả là số thực:/ ví dụ 7/2 cho ta kết quả là 3.5
- Phép chia lấy kết quả là nguyên: DIV

Ví dụ: 13 DIV 2 cho kết quả 6

- Phép chia lấy số dư của hai số nguyên: MOD

Ví dụ: 7 MOD 4; cho ta giá trị bằng 3 là số dư của phép tính .

- Hàm Boolean ODD (n) cho kết quả TRUE nếu n là một số lẻ, FASLE nếu n là số chẵn.

Chú ý: Khi thực hiện các phép tính số học đối với số nguyên, cần quan tâm xem các kết quả có vượt ra khỏi phạm vi biểu diễn số nguyên của máy hay không.

Ví dụ:

31950 +3800= 35750 (vượt quá giới hạn)

2. Các phép tính quan hệ đối với số nguyên:

Các số nguyên có thể so sánh với nhau và cũng có thể so sánh với các số thực. Kết quả của phép toán so sánh là TRUE (đúng) hoặc FALSE (sai).

Ví dụ:

12<4 cho kết quả FALSE

23<67.2345 cho kết quả TRUE.

3. Số nguyên có kiểu BYTE:

Cách biểu diễn số nguyên như trên là cách biểu diễn số nguyên có dấu có độ lớn hai byte và đi với từ khoá Integer. Bên cạnh đó còn có một kiểu đơn giản chuẩn là kiểu biểu diễn số nguyên bằng một byte. Phạm vi của nó từ 0 đến 255.

2.3 Kiểu số thực (Real)

Tương tự như định nghĩa kiểu số nguyên, kiểu số thực là tập hợp các số thực có thể biểu diễn được trong máy tính và được máy định nghĩa sẵn với từ khoá Real.

Các phép toán cộng (+), trừ t (-), nhân n (*), chia (/) cũng như các phép toán quan hệ c (=, <>, < , >, >=, <=) đều có thể áp dụng cho các toán hạng l à số thực lẫn toán hạng là số nguyên.

Tất nhiên không tồn tại các phép toán DIV, MOD cho kiểu số thực.

Các số thực được biểu diễn trong máy ở hai dạng: dạng dấu chấm tĩnh v à dạng dấu chấm động (có phần số mũ).

Dạng dấu chấm tĩnh như sau: (Dấu chấm thay cho dấu phẩy thập phân)

13.23 5.00 -32.3456 -0.00001

Gọi là tĩnh vì dấu chấm thập phân ở đâu đặc trưng cho độ lớn của số đó

Dạng dấu chấm động (có phần mũ):

Gồm hai phần: phần định trị và phần mũ viết sau chữ E để biểu diễn số mũ của cơ số 10.
Ví dụ:
$1233.45 = 1.23345*10^3$
Biểu diễn bằng số dấu chấm động như sau:
1.23345E+03
Gọi là động vì dấu chấm thập phân chưa đủ để xác định độ lớn của số đó.
Cách viết số thực như sau là sai, là không đủ:

.8 (Phải viết là 0.8)
3. (Phải viết là 3.0 hoặc 3)

Các hàm số học chuẩn (dùng cho số nguyên và số thực):
Các hàm sau đây được định nghĩa sẵn và được sử dụng với đối số là các số thực hoặc các số nguyên:

- ABS(x) cho kết quả là trị tuyệt đối của x
- SQR(x) cho giá trị x^2.
- SIN(x), COS(x), ARCTAN(x) là các hàm lượng giác.
- LN(x) hàm log2 (x) =ln(x).
- EXP(x) hàm ex
- SQRT(x) hàm tính căn bậc hai của x
- SUCC(n) đối số nguyên n, cho số nguyên tiếp theo n, tức là n +1.
- PRED(n) đối số nguyên n, cho ta số nguyên trước n, tức là n -1.
- ODD(n) đối số nguyên n, TRUE nếu n lẻ, FALSE nếu n chẵn.
- TRUNC(x) kết quả là phần nguyên của x, tức là cắt bỏ đi phần lẻ thập phân của x. Ví dụ trunc (3.146) = 3.
- ROUND(x) cho ta một số nguyên là phần nguyên của x bằng cách qui tròn phần lẻ thập phân của x. Ví dụ:

Round (6.76) = 7
Round (6.36) = 6
Round (-34.812) = -35

2.4 Kiểu ký tự (character)

Máy tính không chỉ xử lý các dữ liệu bằng số mà còn có khả năng xử lý các dữ liệu văn bản (ký tự).
Kiểu ký tự được định nghĩa với từ khoá CHAR.
Hằng ký tự phải đặt trong mở nháy đơn và đóng nháy đơn, ví dụ:

'5', 'b', 'A', '0', '9', '*', '!',...

Máy tính dùng bộ ký tự để trao đổi thông tin qua các thiết bị vào - ra. Bộ ký tự được dùng phổ biến nhất là bộ ký tự ASCII (American Standard Code for information Interchange). Hay còn gọi là bộ mã ASCII. Các ký tự được mã hoá bằng một byte, một Byte có thể biểu diễn được 256 ký tự khác nhau. Các ký tự cơ bản nhất từ 0 đến 127 được gọi là ASCII chuẩn. Còn từ 128 đến 255 gọi là

ASCII mở rộng và được dùng để mã hoá các ký tự riêng của một số ngôn ngữ, các ký tự toán học, đồ họa. Chúng ta quan tâm chủ yếu tới phần 128 ký tự đầu của bảng mã ASCII để xây dựng cú pháp cho lập trình.

Trong ASCII, các ký tự từ 0 đến 31 là các ký tự điều khiển, không in được, dùng để điều khiển các thiết bị ngoại vi, điều khiển việc trao đổi thông tin... Ví dụ khi máy nhận được ký tự số 7 (BEL), máy sẽ phát ra một tiếng kêu. Khi nhận được ký tự số 13 (CR : Carriage Return) con trỏ màn hình sẽ xuống đầu dòng tiếp theo. Khi nhận được 10 (LF: Line Feed) đầu kim in của máy in sẽ xuống đầu dòng tiếp theo v.v..

Phần còn lại trong ASCII là mã của các chữ cái A, B, C,... các chữ số 0, 1,...,9, các ký tự đặc biệt ... Ví dụ khi nhận được mã 65, máy sẽ hiện lên màn hình chữ cái A.

Riêng mã 127 (Del) được dùng làm ký tự điều khiển xoá. Khi soạn chương trình, phím Delete trên bàn phím chính là phím tạo ra mã số 127 để xoá một ký tự trên màn hình.

Hai hàm chuẩn ORD và CHR cho phép thiết lập tương quan giữa mã ký tự và ký tự tương ứng, ví dụ:

Hàm ORD ('A') cho kết quả 65, số thứ tự (mã) của ký tự 'A' trong bảng mã.

Hàm CHR (65) cho ta ký tự A.

Đương nhiên là CHR (ORD('A')) = 'A' và ORD (CHR(65)) =65

Ký tự Chr1 nhỏ hơn ký tự Chr2 nếu Ord (Chr1) < ord (Chr2).

Hai cách viết sau là tương đương:

#65 = CHR(65)

Hàm chuẩn PRED và SUCC có thể áp dụng cho đối số là ký tự, kết quả là ký tự. Giả sử Chr là một ký tự nào đó:

Hàm PRED (Chr) cho kết quả là một ký tự nằm trước ký tự Chr. Ví dụ PRED ('B')='A'. Hàm SUCC (Chr) cho kết quả là một ký tự nằm sau ký tự Chr, ví dụ: SUCC('A')='B'

2.6 Mô tả số nguyên với Word, Shortint, Longint.

Số nguyên ngoài Integer và Byte còn có với các dạng Word, Shortint (Short Integer), Longint (Long Integer). Sau đây là bảng kiệt kê các phạm vi của nó.

Kiểu	**Phạm vi**	**Kích thước Byte**
Byte	0..255	1
Shortint	-128..127	1
Integer	-32768..32767	2
Word	0..65535	2
Longint	-214783648..21483647	4

Turbo Pascal còn cho phép biểu diễn số nguyên dưới dạng hệ 16 (số Hexa) bằng cách dùng thêm dấu $ ở đầu. Ví dụ: $E có giá trị là 14 (hệ thập phânh).

Một byte được biểu diễn bằng hai chữ số Hexa, vì một chữ số Hexa tương đương với số nhị phân 4 bít. Ta có bảng quan hệ sau:

Hệ 10	**Hệ 2**	**Hệ 16**
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Ví dụ: số Hexa $7FFF =7*16^3+15*16^2+15*16^1+15*16^0= 32767 (s ố hệ 10)

$8AB2=8*16^3+10*16^2+11*16^1+2*16^ 0=35506 (số hệ 10)

2.7 Mở rộng khai báo kiểu số thực

Số thực còn có các dạng sau:

Kiểu	**Phạm vi**	**Số chữ số có nghĩa**	**Kích thước (bytes)**
REAL	2.9E-39.. 1.7E+38	11-12	6
SINGLE	1.5E-45..3.4E+38	7-8	4
DOUBLE	5.0E-324..1.7E+308	15-16	8
EXTENDED	1.9E-4951..1.1E+4932	19-20	10

Chương 3: Khai Báo, biểu thức và câu lệnh

Hằng, biến, kiểu đều phải được khai báo trước khi dùng trong chương trình.

3.1 Khai báo hằng

Có các loạiC: hằng nguyên, hằng thực, hằng ký tự và hằng Boolean.
Khai báo hằng: Các hằng được khai báo như sau:
Const tên_hằng = giá_trị_của_hằng;
Một dòng khai báo hằng được kết thúc bằng dấu chấm phẩy. Tên hằng được đặt theo qui tắc đã trình bày ở chương trước.
Ví dụ:

```
Const
    X=5;            (*Hằng nguyênH *)
    Y=3.14;         (*Hằng thực 5H*)
    A=TRUE;         (*Hằng BooleanH *)
    Chr= 'e';       (*Hằng ký tự eH *).
```

Các hằng không thay đổi trong suốt chương trình.

3.2 Khai báo biến

Biến (variable) là đại lượng có thể thay đổi giá trị trong quá trình xử lý. Tên biến của chương trình đồng thời là địa chỉ của biến đó. Biến được khai báo như sau:

```
Var
    Tên_biến: Kiểu_dữ_liệu_của_biến;
```

Dấu hai chấm để ngăn cách hai phần của khai báo, dấu chấm phẩy kết thúc một dòng khai báo.
Nếu có nhiều biến cùng kiểu có thể được rút gọn bằng cách viết tên các biến đặt cách nhau dấu phẩy (,) ví dụ:

```
Var
    a, b, c: Real;
    i,j,k: Integer;
    T: Boolean;
```

3.3 Định nghĩa kiểu

Đối với các dữ liệu cơ sở (Real, Integer, Byte, Char, Boolean) đã được định nghĩa sẵn nên ta khai báo biến trực tiếp ngay. Còn các kiểu dữ liệu khác ta phải định nghĩa kiểu.
Ví dụ:

```
Type
    Mau = (Xanh, Do, Vang);
Var
    MauSac: Mau;
```

3.4 Biểu thức (expression)

Một biểu thức gồmM: toán tử (operator) và toán hạng (operand). Toán tử (dấu phép toán), toán hạng có thể là hằng, hàm, biến.
Ví dụ: 14 + SQRT(5)-2*COS(X)
Trong ví dụ này, các phép toán là phép cộng (+), phép trừ p (-) và phép nhânv (*). Các toán hạng là hằng số 14, hàm SQRT với đối số là 5, hằng 2 và hàm COS với đối số là biến X .
Biểu thức số học là biểu thức có giá trị bằng số (là integerl, byte, real)
Biểu thức Boolean là biểu thức có giá trị là TRUE hoặc FALSE.
Một biểu thức chứa các toán tử quan hệ (<>, >, <=, >=, =) cũng được gọi là biểu thức Boolean hay một biểu thức quan hệ. Các toán hạng trong biểu thức quan hệ có thể là nguyên, thực, ký tự v.v..
Thứ tự thực hiện từ trái qua phải với thứ tự ưu tiên như sau:

Dấu ngoặc ( )	Biểu thức trong ngoặc được ưu tiên thứ nhất
Not, - (dấu trừ)	Các phép toán một toán hạng
*, /, Div, mod, and	Các phép tính loại nhân, cùng mức ưu tiên
+, -, or, xor	Các phép tính loại cộng, cùng mức ưu tiên
=, <>, <=, >=, >, < In	các phép toán quan hệ, có cùng mức ưu tiên

Các ví dụ sau minh hoạ về thứ tự tính toán.
Ví dụ:
2 + 4*3 = 2 +(4*3) =2+12=14
3/2*4 = (3/2)*4 = 1.5*4= 6.0
9 Div 2 *4 = (9 div 2) * 4 = 16
(2>3) and ('c' < 'd')=FALSE AND TRUE=FALSE
Not(3 + 1 < 5) OR (3 > 5 Div 3)=Not (4<5) OR (3>1)=Not TRUE OR TRUE=FALSE OR TRUE=TRUE

3.5 Câu lệnh (Instruction, Statemment)

Ngoài phần mô tả dữ liệu còn có phần lệnh (instruction) của chương trình. Phần này quy định các công việc mà chương trình phải thực hiện để xử lý các dữ liệu. Các lệnh (hay cũng gọi là câu lệnh) cách nhau bởi dấu chấm phẩy. Câu lệnh được chia ra hai loại: Câu lệnh đơn giản và câu lệnh có cấu trúc.

- Câu lệnh đơn giản bao gồm: Lệnh gán (:=), lời gọi thủ tụcl, lệnh nhẩy GOTO
- Câu lệnh có cấu trúc bao gồm:
  Câu lệnh ghép (lệnh hợp thànhl): BEGIN ...END;
  Câu lệnh điều kiện và lựa chọn: if .. then... else... case... of...

Các vòng lặp: For... To... Do...
Repeat ...Until...
While... Do...
Với....làm : With...Do

Mỗi một câu lệnh bao giờ cũng được đặt cách nhau bằng dấu chấm phẩy (;). Dấu chấm phẩy (;) chỉ có tác dụng ngăn cách các câu lệnh chứ không phải là dấu kết thúc câu lệnh và nó không thuộc vào câu lệnh. Pascal không bắt buộc phải viết mỗi câu lệnh một dòng. Vấn đề là phải trình bầy chương trình sao cho đẹp, rõ ràng, mô tả được thuật toán...

Ví dụ: Chúng ta có thể viết liền nhau trên một dòng chương trình:

```
A='Chao anh'; B:=TRUE;C :=12;
```

Hoặc viết thành từng dòng:

```
A='Chao anh';
B:=TRUE;
C:=12;
```

Các lệnh có cấu trúc sẽ được đề cập ở các chương sau. Còn thủ tục sẽ được đưa vào cùng với các cấu trúc dữ liệu.

- **Phép gán**

Phép gán dùng để gán giá trị của một biểu thức, một hằng vào một biến. Phép gán được viết như sau:

biến:= biểu_thức;

Ví dụ:

A:=12; biến A nhận giá trị bằng 12 b (hay đưa 12 vào biến có địa chỉ A)

B:= FALSE; biến B nhận giá trị FALSE.b

Y:= Y+1 Y cộng với 1 kết quả để tại Y

Khi dùng phép gán thì kiểu của biến và kiểu của biểu thức ở vế phải giống nhau. Ví dụ một biến thực X không thể nhận một giá trị Boolean được.

X := TRUE là sai.

Trường hợp ngoại lệ: một biến nguyên không thể nhận một giá trị là số thực nhưng một biến thực thì lại có thể nhận một giá trị nguyên. Giả sử I là Integer. I:= 9.5 là sai mà phải viết I:=Round(9.5). Nhưng nếu X là biến thực, viết như sau là đúng:

I:=12; X:=I;

3.6 Lệnh hợp thành hay lệnh ghép

Một nhóm câu lệnh đơn giản được đặt giữa hai chữ Begin và End sẽ tạo thành một câu lệnh hợp thành hay một lệnh ghép với cách mẫu viết như sau:

```
Begin
    Câu lệnh 1;
    Câu lệnh 2;
    ....
```

```
        Câu lệnh n;
    End;
```

Cặp **Begin ... End** của câu lệnh ghép cho ta thấy rõ hơn tính có cấu trúc của ngôn ngữ Pascal. Ta có thể nhóm các lệnh thành từng khối. Một khối lệnh chỉ có một mở đầu bằng chữ **Begin** và một kết thúc bằng chữ **End**. Chúng ta sẽ hiểu chi tiết thêm về lệnh ghép thông qua các ví dụ cụ thể ở các chương tiếp theo.

Bài tập:

Nêu quy tắc đặt tên biến trong chương trình?
Hằng là gì? và được khai báo như thế nào?
Hãy khai báo một biến thực, nguyên bất kỳ?
Biểu thức là gì, cho ví dụ
Thế nào là biểu thức Logic? Cho ví dụ
Thế nào là biểu thức quan hệ? cho ví dụ
Hãy cho một số ví dụ về lệnh gán
Lệnh hợp thành là gì?
Có thể có các lệnh hợp thành nằm trong một lệnh hợp thành? Cho ví dụ
Kết quả của biến A là gì nếu sau khi thực hiện lệnh gán sau:
A:=((3<1) AND (12>3)) OR (4<11)

Chương 4: Thủ tục vào, ra dữ liệu.

4.1 Thủ tục hiển thị dữ liệu ra màn hình: Write và Writeln

Để hiển thị dữ liệu ra màn hình, Pascal có 3 cách viết như sau:

Write (Item1, Item2,..., ItemN);
Writeln(Item1, Item 2,..., ItemN); (đọc là write line)
Writeln;

Trong đó (Item1, Item2,..., ItemN); là các mục cần hiển thị, có thể là một trong các loại sau:

Ví dụ:

- Hằng: Writeln(3.12); Writeln(' Chào bạn!');
- Biến: Writeln(A);
- Hàm: Writeln(Cos(x));
- Biểu thức: Writeln(X+Y - 2 +Exp(X));

Một số kiểu có cấu trúc như mảng, xâu ký tự (sẽ học saus).

Màn hình thông dụng hiện nay được phân thành 25 dòng, mỗi dòng có thể chứa 80 ký tự.? mỗi thời điểm, tại một vị trí xác định trên màn hình có một khối sáng nhấp nháy hoặc một nét gạch nhấp nháy gọi là ***con trỏ màn hình*** hay điểm nhắc (cursor) .

Sự khác nhau của 3 cách viết trên là ở chỗ vị trí con trỏ màn hình sau khi kết thúc lệnh.

- **Cách viết Writeln (Item1, Item2,..., ItemN); có ý nghĩa:**

Sau khi hiển thị các giá trị của (Item1, Item2, ..., ItemN) trên cùng một dòng, con trỏ sẽ chuyển xuống dòng tiếp theo (con trỏ được đặt ở đầu dòng tiếp theo)

Cách viết Write (Item1, Item2,..., ItemN);

Sau khi hiển thị các giá trị của (Item1, Item2,..., ItemN) con trỏ sẽ không chuyển xuống đầu dòng tiếp theo (con trỏ sẽ được đặt ở vị trí sau giá trị của ItemNc).

Cách viết Write (Item1, Item2,..., ItemN); là tương đương với khối lệnh như sau:

```
Begin
    Write(Item1);
    Write(Item2);
    ...
    Write(ItemN);
End;
```

Thủ tục Writeln; (không có tham sốk) sẽ đặt con trỏ xuống đầu dòng tiếp theo. Vì vậy

Writeln(Item1, Item2,..., ItemN); tương đương với một lệnh ghép.

```
Begin
    Write( Item1, Item2,..., ItemN);
    WriteLn;
```

```
End;
```

Trong khi dùng thủ tục **Write** và **Writeln**, có hai cách trình bày dữ liệu in ra là cách viết không qui cách và cách viết có qui cách. Chúng ta sẽ xét cụ thể lệnh viết ra với từng kiểu dữ liệu.

1. Viết ra kiểu số nguyên.

Cách viết không qui cách:

```
Var
   I : Integer;
   Begin
      I :=345;
      Writeln(I);
      Writeln(-98671);
   End.
```

Sẽ hiển thị ra màn hình:

```
345
-98671
```

Cách viết không qui cách không chỉ rõ qui cách hiển thị trong các thủ tục Write. Các số nguyên sẽ được viết ra với số chỗ bằng độ dài của nó (số chữ số của nó).? đây biến I có giá trị 345 chiếm 3 vị trí, còn -98671 chiếm 6 vị trí (5 chữ số và dấu 5 -) kể từ vị trí bên trái.

- **Cách viết có qui cách.**

Ta có thể bố trí chỗ số cố định để hiển thị số nguyên bằng cách thêm dấu hai chấm (:) theo sau là số vị trí cần hiển thị. Máy sẽ xếp chỗ từ bên phải qua bên trái. Nếu thừa sẽ để trống ở bên trái.

Ví dụ:

```
Writeln (' 123 456');
I:= 345;
Writeln(I:6);
Writeln(-98671:8);
```

Sẽ viết ra trên màn hình:

```
123456
   345
 -30567
```

Như vậy cách viết không qui cách sẽ căn lề bên trái còn cách viết có qui cách sẽ căn lề bên phải.

2. Viết ra kiểu số thực

Với khai báo X là một biến thực

```
Var
   X: Real;
```

- **Cách viết không qui cách**

Sẽ hiển thị giá trị của X theo cách biểu diễn dấu chấm động, ví dụ:

```
X :=34.56;
```

```
Writeln(X);
Write(6.72);
```

sẽ cho kết quả trên màn hình như sau:

```
3.4560000000E+01
6.7200000000E+00
```

Hai dấu cách ở đầu, một số phần nguyên, dấu chấm thập phân, 10 chỗ phần thập phân
(nếu không có giá trị cụ thể thì sẽ bị điền số 0 vào), chữ E biểu diễn số mũ, dấu của phần mũ, hai số biểu diễn giá trị phần mũ. Tổng số chỗ số thực chiếm là 18.

- **Cách viết có qui cách của số thực X:**

Cách 1: Hiển thị số dấu chấm tĩnh

X : < số chỗ cho cả số thực >:< số chỗ cho phần thập phân >

Cách 2: Hiển thị số dấu chấm động

X :< số chỗ cho viết dưới dạng E mũ >.

Ví dụ: áp dụng cho cách 1

```
X:=123.456;
Writeln(25.123456789:12:6);
Writeln(X :12:6);
Writeln( 3.14:12:6);
```

cho ra kết quả:

```
25.123457
123.456000
 3.140000
```

Phần thập phân được xác định 6 chỗ, nếu không đủ giá trị cụ thể, nó sẽ điền số 0 vào. Còn nếu phần thập phân mà nhiều, nó sẽ làm tròn như dòng đầu tiên (số 25.123457s).

Với cách viết có qui cách này, ta có thể trình bày các số thực sao cho các dấu chấm thập phân thẳng theo chiều đứng một cách đẹp mắt.

Ví dụ: áp dụng cho cách 2.

```
Write ( 25.123456789:12);          (cho ra kết quả có làm tròn)
      2.512346E +01
```

Trong cách 2, số thực được viết ra dưới dạng E mũ (phần mũ chiếm 4 chỗp), toàn bộ số chỗ là 12 thì phần còn lại để viết phần định trị là 12-4=8 chỗ.

3. Hiển thị kiểu ký tự:

Với cách viết không qui cách, mỗi ký tự chiếm một chỗ. Còn với cách viết có qui cách, máy sẽ bố trí thêm các chỗ trắng.

```
Begin
  Writeln('ABCD');          (*Cách viết không có qui cáchC *)
  Writeln('E');             (*Cách viết không có qui cáchC *)
  Writeln('E':5);           (*Cách viết có qui cáchC *)
  Writeln('ABCD' :5);       (*Cách viết có qui cáchC *)
End.
```

sẽ cho ra kết quả

ABCD
E

ABCD
E

4. Viết ra kiểu Boolean

Kiểu Boolean sẽ được hiển thị với các từ **True** và **False**. Ta cũng có thể qui định số chỗ để viết các từ này.

Ví dụ:

```
Var
    a: Boolean;
Begin
    a: =5>12;
    Writeln(a);          (*Cách viết không qui cách C *)
    Writeln(a:7);        (*Cách viết có qui cách C *)
End.
```

sẽ cho ra kết quả:

FALSE
FALSE

5. In ra máy in

Write và Writeln được dùng để hiển thị dữ liệu ra màn hình, muốn in ra giấy (sử dụng máy in) ta thêm từ **Lst** và dấu phẩy vào trước các biến hoặc hằng cần in ra.

Ví dụ:

```
Writeln (Lst, X:5:2, Y:12:5, I:5);
```

Ngoài việc phải thêm **Lst** còn phải khai báo:

```
USES PRINTER;
```

6. Trình bày màn hình:

Để trình bày màn hình theo những thủ tục xác định, ở phần khai báo phải có:

USES CRT;

CRT là đơn vị (Unit) chứa các thủ tục trình bày màn hình

- **GOTOXY**(X,Y);

Là thủ tục đưa con trỏ đến toạ độ X, Y của màn hình. X là tọa độ cột tính từ 1 đến 80, Y là tọa độ dòng, tính từ 1 đến 25 (màn hình được chia thành 25 dòng, 80 cột).

Ví dụ: GOTOXY(15,14); (đưa con trỏ đến cột 15 và dòng 14)

- **ClrScr**; (Clear Screen) là thủ tục xoá toàn bộ màn hình, sau khi xoá con trỏ nằm ở vị trí góc trái phía trên màn hình.

ClrEol; (Clear end of Line) thủ tục xoá các ký tự nằm bên phải con trỏ đến hết dòng.

TextColor( ) thủ tục xác định màu chữ, ví dụ Textcolor (Blue) sẽ cho chữ màu xanh.

- **TextBackground( )** là thủ tục xác định màu nền của văn bản, ví dụ: TextBackGround(Red) sẽ cho màu nền đỏ.

NormVideo: Đặt văn bản có độ sáng bình thường

LowVideo: Đặt văn bản có độ sáng yếu

Ta có bảng màu sau:

Tên màu	Tiếng Việt	Mã
Black	Đen	0
Blue	Xanh lơ	1
Green	Xanh lá cây	2
Cyan	Xanh cẩm thạch	3
Red	Đỏ	4
Magenta	Tía	5
Brown	Nâu	6
LightGray	Xám nhạt	7
DarkGray	Xám đậm	8
LightBlue	Xanh nhạt	9
LightGreen	Xanh lá cây nhạt	10
LightCyan	Xanh cẩm thạch nhạt	11
LightRed	Đỏ nhạt	12
LightMagenta	Tía nhạt	13
Yellow	Vàng	14
White	Trắng	15
Blink	Nhấp nháy	128

Hai cách viết: Textcolor(Blue) và Textcolor (1) là tương đương.

Ví dụ 1:

```
USES CRT;
Var
   X,Y: Integer;
Begin
      TextBackground (Red); (*Đặt mầu nền cho màn hình *)
      ClrScr;       (*Xoá màn hình *)
      X :=6; Y :=3;
      GotoXY (X,Y);TextColor(Blue);
      Writeln ('Chao ban'); (*Chữ Chao ban sẽ được hiện ở toạ độ (6,3) và màu xanh *)
      GotoXY (X+10,Y+2);     TextColor(White);
      Writeln ('Chao ban');(* Chữ Chao ban được hiện ở toạ độ (16,5) và màu trắng *)
      TextColor(White + Blink);
```

Write ('Chao ban'); (*Chữ Chao ban hiện ở toạ độ (1,6), màu trắng nhấp nháy *)

End.

Ví dụ 2:

Hiển thị từ COMPUTER (với chữ CO và ER sáng bình thường còn lại là sáng yếu)

```
USES CRT;
    Begin
    NormVideo; write('CO');
    LowVideo;  write('MPUT');
    NormVideo; write('ER');
    End.
```

Chú ý: Nếu bạn chạy Turbo Pascal 7.0 với một số CPU có tốc độ cao thì Crt sẽ gây lỗi trong một số trường hợp khi gặp thủ tục Writeln. Tuỳ theo từng trường hợp cụ thể mà có thể có cách khắc phục. Borland sẽ không cập nhật tiếp để nâng cấp dòng Turbo Pascal.

4.2 Vào dữ liệu: READ và READLN

1. Read và Readln:

Nhờ phép gán ta có thể đưa giá trị cho biến (từ một biến khác hay hằng) ở trong chương trình. Vậy muốn đưa một giá trị từ bàn phím (thông qua gõ và đưa vào biến) thì làm thế nào? Thủ tục **Read** và **Readln** (Read Line) sẽ giúp ta làm điều đó. Ba cách viết sau:

READ (Var1, Var2,...,VarN);
READLN(Var1, Var2,...,VarN);
READLN; (Không có tham số)

Trong đó Var1, Var2... là các biến (không thể hằng) đã khai báo trong chương trình.

Khi gặp thủ tục này chương trình sẽ chờ để chúng ta gõ dữ liệu vào. Dữ liệu gõ vào phải có cùng kiểu với biến, các dữ liệu ngăn cách ít nhất một khoảng trống (Space), sau khi gõ xong dữ liệu hãy gõ tiếp ↵ (Enter)**.**

Ví dụ 1:

```
Var
    i,j: Integer;
    a,b,c: Real;
Begin
    Readln (i,j);
    Readln(a,b,c);
    .....
End.
```

Khi chạy chương trình (sẽ nói ở phần sau) máy sẽ chờ để chúng ta gõ 2 số nguyên ứng với thủ tục Readln (i,j), ví dụ: 4 6 ↵ (Enter). Máy lại chờ để ta gõ