TRƯỜNG CAO ĐẲNG NGHỀ CÔNG NGHIỆP HÀ NỘI
Tác giả: Trần Thị Ngân (chủ biên).
Lê Văn Hùng

GIÁO TRÌNH
Lập trình căn bản
(Lưu hành nội bộ)

Hà Nội năm 2012

Tuyên bố bản quyền

Giáo trình này sử dụng làm tài liệu giảng dạy nội bộ trong trường cao đẳng nghề Công nghiệp Hà Nội

Trường Cao đẳng nghề Công nghiệp Hà Nội không sử dụng và không cho phép bất kỳ cá nhân hay tổ chức nào sử dụng giáo trình này với mục đích kinh doanh.

Mọi trích dẫn, sử dụng giáo trình này với mục đích khác hay ở nơi khác đều phải được sự đồng ý bằng văn bản của trường Cao đẳng nghề Công nghiệp Hà Nội

LỜI GIỚI THIỆU

Giáo Trình Lập trình căn bản Pascal được biên soạn nhằm đáp ứng yêu cầu học tập của học sinh, sinh viên bước đầu làm quen với công việc lập trình, đồng thời 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 .

Giáo trình không chỉ phù hợp cho người mới bắt đầu mà còn phù hợp cho những người cần tham khảo. Nội dung của giáo trình được chia thành 6 chương:

Chương 1: Làm quen ngôn ngữ lập trình
Chương 2: Các thành phần trong ngôn ngữ lập trình
Chương 3: Các cấu trúc điều khiển
Chương 4: Hàm và thủ tục
Chương 5: Dữ liệu kiểu tập hợp, mảng và bản ghi
Chương 6: Dữ liệu kiể chuỗi

Khi biên soạn, chúng tôi đã tham khảo các giáo trình và tài liệu giảng dạy môn học này của một số trường Cao đẳng, Đại học để giáo trình vừa đạt yêu cầu về nội dung vừa thích hợp với đối tượng là sinh viên của các trường Cao đẳng Nghề.

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.

MỤC LỤC

MÔN HỌC LẬP TRÌNH CĂN BẢN

Mã số của môn học: MH15
Vị trí, ý nghĩa, vai trò môn học/mô đun:

- Vị trí: Môn học được bố trí sau khi sinh viên học xong các môn học chung, các môn học tin đại cương, tin văn phòng.
- Tính chất : Là môn học lý thuyết cơ sở nghề bắt buộc.

Mục tiêu của môn học/mô đun:

- Trình bày được khái niệm về lập máy tính;
- Mô tả được ngôn ngữ lập trình: cú pháp, công dụng của các câu lệnh;
- Phân tích được chương trình: xác định nhiệm vụ chương trình;
- Thực hiện được các thao tác trong môi trường phát triển phần mềm: biên tập chương trình, sử dụng các công cụ, điều khiển, thực đơn lệnh trợ giúp, gỡ rối, bẫy lỗi,v.v.;
- Viết chương trình và thực hiện chương trình trong máy tính.
- Bố trí làm việc khoa học đảm bảo an toàn cho người và phương tiện học tập.

Mã bài	Tên chương mục/bài	Loại bài dạy	Địa điểm	Thời lượng			
				Tổng số	Lý thuyết	Thực hành	Kiểm tra
MH15-01	Làm quen ngôn ngữ lập trình	Lý thuyết	Lớp học	5	5	0	-
MH15-02	Các thành phần cơ bản trong ngôn ngữ lập trình	Tích hợp	Phòng thực hành	15	5	9	1
MH15-03	Các cấu trúc điều khiển	Tích hợp	Phòng thực hành	25	10	14	1
MH15-04	Hàm và thủ tục	Tích hợp	Phòng thực hành	25	10	14	1
MH15-05	Dữ liệu kiểu tập hợp, mảng và bản ghi	Tích hợp	Phòng thực hành	30	10	19	1
MH15-06	Dữ liệu kiểu chuỗi	Tích hợp	Phòng thực hành	20	5	14	1

YÊU CẦU VỀ ĐÁNH GIÁ HOÀN THÀNH MÔN HỌC/MÔ ĐUN

- Về kiến thức: Được đánh giá qua bài kiểm tra viết, trắc nghiệm đạt được các yêu cầu sau:

- Vận dụng quy tắc cú pháp của ngôn ngữ, các hoạt động vào/ra, tuần tự và tuyến tính.

- Xác định các điều khiển áp dụng cho việc nhập dữ liệu đảm bảo chính xác, có chu trình xử lý dữ liệu.

- Mô tả chức năng và viết chương trình logic (pseudo code) của từng mô đun xử lý của hệ thống.

- Vận dụng các phương pháp lặp điều kiện trước hoặc sau, đảm bảo điều kiện kết thúc của vòng lặp.

- Về kỹ năng: Đánh giá kỹ năng thực hành của sinh viên trong bài thực hành Lập trình căn bản đạt được các yêu cầu sau:

- Xác định môi trường hoạt động của hệ thống (các điều khiển, công cụ, các thành phần, tập hợp dữ liệu, nhập dữ liệu, in kết quả ...)

- Chú thích cho từng đoạn xử lý của chương trình.

- Về thái độ: Đánh giá tính tự giác, tính kỷ luật, tham gia đầy đủ thời lượng môn học, cẩn thận, tỉ mỉ, chính xác trong công việc.

CHƯƠNG 1:

Tên chương: LÀM QUEN NGÔN NGỮ LẬP TRÌNH
Mã chương:MH15-01
Mục tiêu:

- Trình bày được các khái niệm về lập trình;
- Trình bày được lịch sử phát triển, ứng dụng của ngôn ngữ lập trình;
- Làm quen môi trường phát triển phần mềm;
- Sử dụng được hệ thống trợ giúp từ help file.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung:

1.1. KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

Lập trình là sử dụng cấu trúc dữ liệu và các câu lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán.

Những ngôn ngữ lập trình (programming language) đầu tiên trên máy tính điện tử là ngôn ngữ máy (machine language), tổ hợp của các con số hệ nhị phân, hay các *bit* (**b**inary dig**it**) 0 và 1. Ngôn ngữ máy phụ thuộc vào hoàn toàn kiến trúc phần cứng của máy tính và các quy ước khắt khe của nhà chế tạo. Để giải các bài toán, những người lập trình phải sử dụng một tập hợp các lệnh điều khiển rất sơ cấp mà mỗi lệnh là tổ hợp các bit nhị phân nên gặp rất nhiều khó khăn, mệt nhọc, rất dễ gặp phải sai sót, nhưng rất khó sửa lỗi.

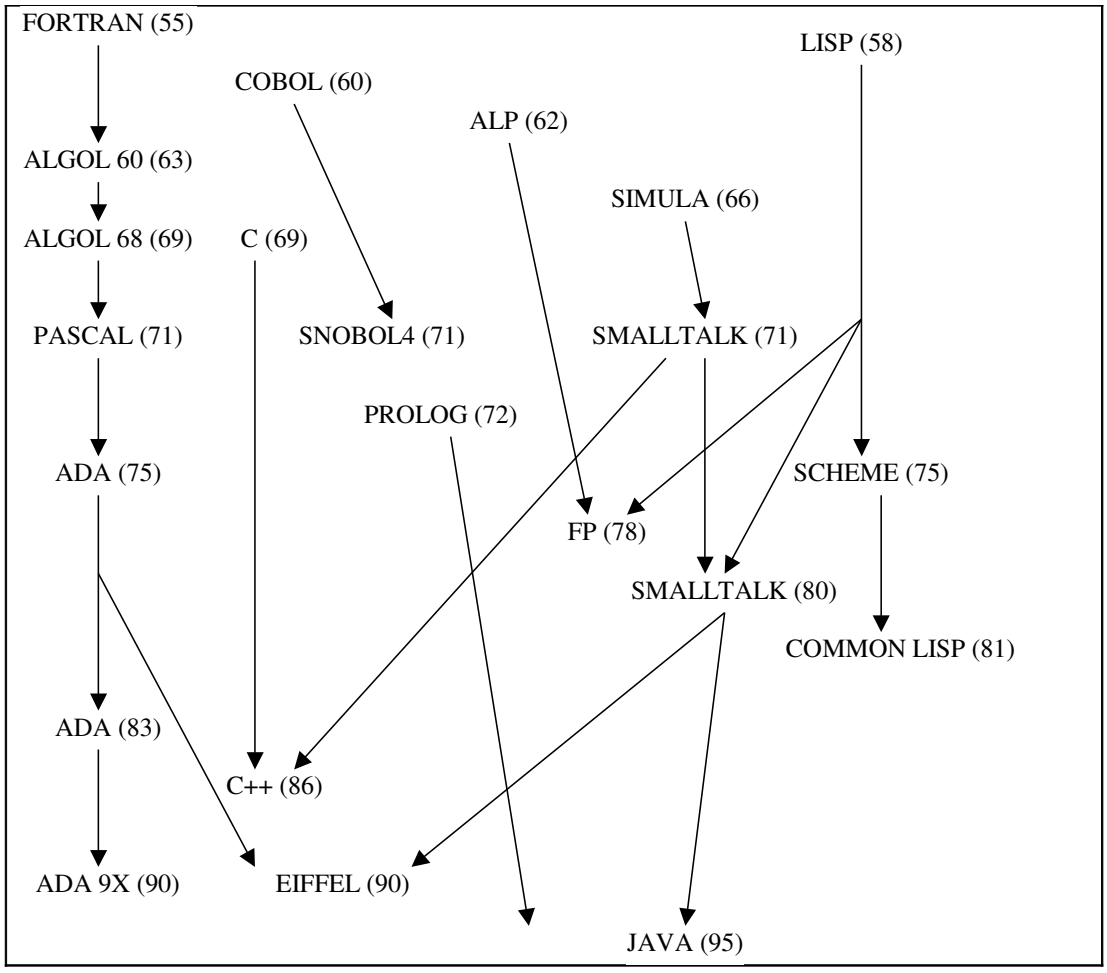
Từ những năm 1950, để giảm nhẹ việc lập trình, người ta đưa vào kỹ thuật *chương trình con* (sub-program hay sub-routine) và xây dựng các *thư viện chương trình* (library) để khi cần thì gọi đến hoặc dùng lại các đoạn chương trình đã viết.

Như thế, chúng ta nhận thấy ở vào giai đoạn sơ khai ban đầu của máy tính điện tử, việc sử dụng máy tính là rất khó khăn, vì ngôn ngữ lập trình là phương tiện giao tiếp lại quá phức tạp đối với người sử dụng. Người sử dụng máy tính vào giai đoạn này chỉ là các chuyên gia về tin học. Như thế, ứng dụng của máy tính điện tử vẫn còn rất hạn chế.

1.2. LỊCH SỬ PHÁT TRIỂN VÀ ỨNG DỤNG CỦA NGÔN NGỮ LẬP TRÌNH PASCAL

Vào đầu những năm 1970 do nhu cầu học tập của sinh viên, giáo sư Niklaus Writh - Trường Đại Học Kỹ Thuật Zurich - Thụy Sĩ đã sáng tác một ngôn ngữ lập trình cấp cao cho công tác giảng dạy sinh viên. Ngôn ngữ được đặt tên là PASCAL để tưởng nhớ đến nhà toán học người Pháp Blaise Pascal.Pascal là một ngôn ngữ lập trình có cấu trúc thể hiện trên 3 phương diện.

- Về mặt dữ liệu: Ngoài các kiểu dữ liệu đơn giản còn có các kiểu dữ liệu có cấu trúc. Ta có thể xây dựng các kiểu dữ liệu phức tạp từ các kiểu dữ liệu đã có.
- Về mặt câu lệnh: Từ các câu lệnh đơn giản và lệnh có cấu trúc ta có thể xây dựng các câu lệnh hợp thành.
- Về mặt chương trình: Một chương trình có thể chia làm nhiều chương trình con



Cho đến nay có hàng trăm ngôn ngữ lập trình được đề xuất nhưng trên thực tế chỉ có một số ít ngôn ngữ được sử dụng rộng rãi. Ngoài cách phân loại theo *bậc* như đã nói ở trên, người ta còn phân loại ngôn ngữ lập trình theo phương thức (paradgm), theo mức độ quan trọng, theo thế hệ, ...

Cách phân loại theo mức hay bậc là dựa trên mức độ trừu tượng so với các yếu tố phần cứng, chẳng hạn như *lệnh* (instruction) và *cấp phát bộ nhớ* (memory allocation) dưới đây:

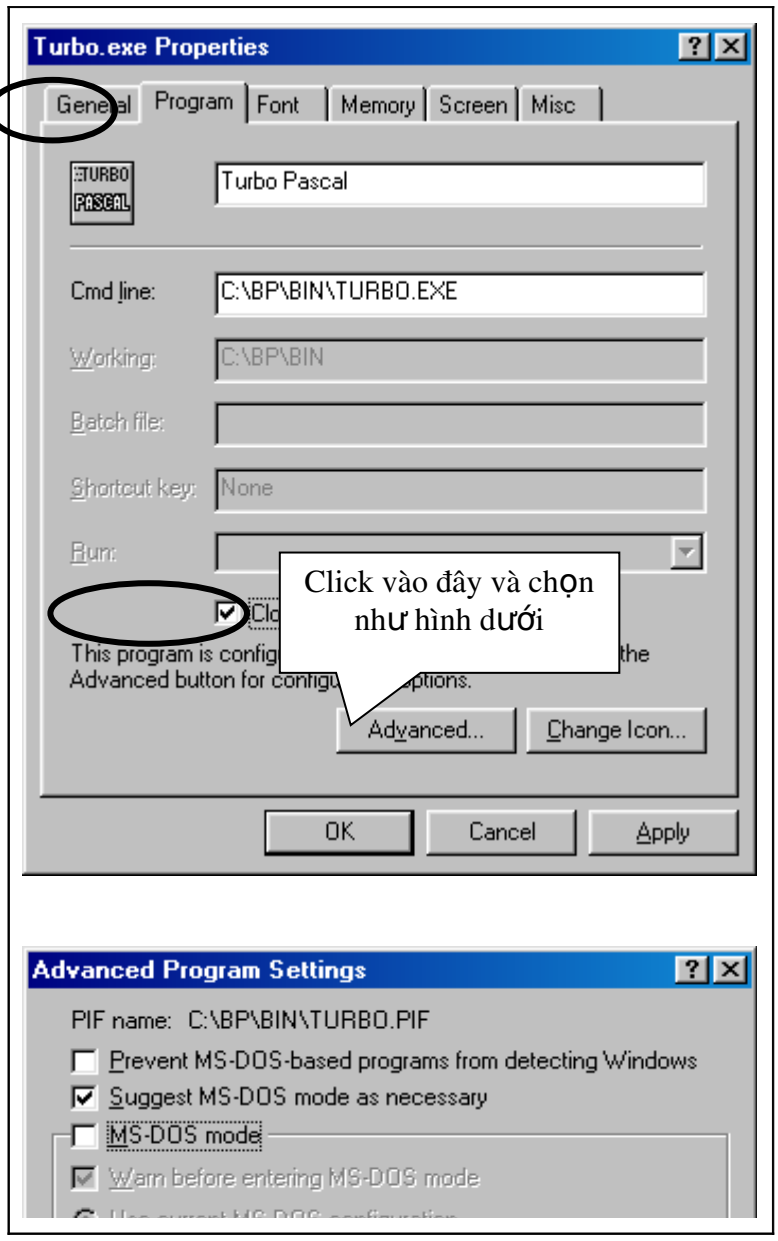
Mức	*Lệnh*	*Sử dụng bộ nhớ*	*Ví dụ*
Thấp	Lệnh máy đơn giản	Truy cập và cấp phát trực tiếp	Hợp ngữ
Cao	Biểu thức và điều kiện tương minh	Truy cập và cấp phát nhờ các phép gán	C, Pascal, Ada
Rất cao	Máy trừu tượng	Truy cập ẩn và tự động cấp phát	Prolog, Miranda

1.3. LÀM QUEN VỚI MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM

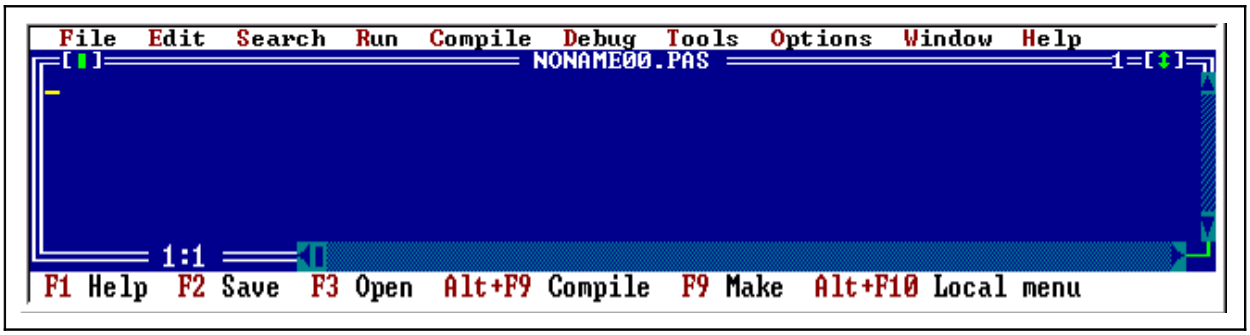
1.4. KHỞI ĐỘNG CHƯƠNG TRÌNH PASCAL

Nếu máy tính chúng ta đã cài đặt Turbo Pascal trên đĩa, ta có thể khởi động chúng như sau (Nếu máy tính chưa có, chúng ta phải cài đặt Turbo Pascal sau đó mới thực thi được)

- Từ MS-DOS: Đảm bảo rằng thư mục hiện hành đúng vị trí cài đặt (hoặc dùng lệnh PATH) Turbo Pascal. Ta đánh vào **TURBO** rồi Enter.
- Từ Windows: Ta nên giả lập MS-DOS Mode cho tập tin **TURBO.EXE** hoặc Shortcut của nó, nếu không mỗi khi ta thực thi TURBO PASCAL chương trình sẽ thoát khỏi Windows, trở về MS-DOS. Sau khi thoát Turbo Pascal ta phải đánh lệnh EXIT để khởi động lại Windows. Cách giả lập như sau:
  - Nhắp chuột phải lên tập tin TURBO.EXE hoặc Shortcut của nó, chọn Properties.
  - Chọn thẻ Program và đánh check như hình sau.



Chọn OK trên các hộp thoại, sau đó khởi động Turbo Pascal, màn hình soạn thảo sau khi khởi động TURBO PASCAL như dưới đây xuất hiện.



1.5. CÁC THAO TÁC SỬ DỤNG TRÊN TURBO PASCAL

Khi ta muốn **tạo mới hoặc mở** một tập tin đã có trên đĩa ta dùng phím **F3**. Sau đó đưa vào tên và vị trí của tập tin. Nếu tập tin đã tồn tại thì Turbo Pascal mở nội dung lên cho ta xem, nếu tên tập tin chưa có thì Turbo Pascal tạo một tập tin mới (với tên mà ta đã chỉ định).

Khi muốn **lưu lại** tập tin ta dùng phím **F2**. Trước khi thoát khỏi chương trình, ta nên lưu tập tin lại, nếu chưa lưu chương trình sẽ hỏi ta có lưu tập tin lại hay không. Nếu ta chọn **Yes** (ấn phím Y) thì chương trình sẽ **lưu lại**, chọn **No** (ấn phím N)chương trình sẽ **không lưu**.

Một số phím thông dụng của TURBO PASCAL 7.0

Biểu tượng	Tên phím	Diễn giải
↵	Enter	Đưa con trỏ xuống dòng.
	Up	Đưa con trỏ lên 1 dòng.
	Down	Đưa con trỏ xuống 1 dòng.
	Left	Đưa con trỏ qua trái một ký tư.
	Right	Đưa con trỏ qua phải một ký tự.
Home	Home	Đưa con trỏ về đầu dòng.
End	End	Đưa con trỏ về cuối dòng.
Pg Up	Page Up	Lên một trang màn hình.
Pg Down	Page Down	Xuống một trang màn hình.
Del	Delete	Xoá ký tự tại vị trí con trỏ.
Back	BackSpace	Xoá ký tự trước con trỏ.
Insert	Insert	Thay đổi chế độ viết xen hay viết chồng.
F1	F1	Gọi chương trình giúp đở.
F2	F2	Lưu tập tin lại.
F3	F3	Tạo mới hoặc mở tập tin.
F4	F4	Thực thi chương trình đến dòng chứa con trỏ.
F5	F5	Phóng lớn cửa sổ.

F6	F6	Chuyển đổi các cửa sổ.
F7	F7	Chạy từng dòng lệnh (hàm xem như một lệnh).
F8	F8	Chạy từng dòng lệnh đơn.
F9	F9	Kiểm tra lỗi chương trình.
Tổ hợp	Alt + F9	Biên dịch chương trình.
Tổ hợp	Ctrl + F9	Chạy chương trình.
Tổ hợp	Ctrl + N	Thêm 1 dòng trước con trỏ.
Tổ hợp	Ctrl + Y	Xoá một dòng tại con trỏ.
Tổ hợp	Ctrl + K + B	Đánh dấu đầu khối.
Tổ hợp	Ctrl + K + K	Đánh dấu cuối khối.
Tổ hợp	Ctrl + K + C	Sao chép khối.
Tổ hợp	Ctrl + K + V	Di chuyển khối.
Tổ hợp	Ctrl + K + Y	Xoá khối.
Trong Borland Pascal các thao tác khối đơn giản và dễ hơn như sau: + Đánh dấu khối: SHIFT + (phím mũi tên) + Copy khối vào clipboard: CTRL+ Ins (phím Insert) + Dán khối (đã copy vào clipboard) vào vị trí mới: SHIFT+ Ins		
Tổ hợp	Ctrl + K + W	Ghi khối lên đĩa thành một tập tin (nội dung của tập tin là khối đã chọn).
Tổ hợp	Ctrl + K + R	Xen nội dung một tập tin (từ đĩa) vào sau vị trí con trỏ.
Tổ hợp	Ctrl + K + H	Tắt/Mở đánh dấu khối.
Tổ hợp	Ctrl + F4	Kiểm tra giá trị biến khi chạy chương trình.
Tổ	Alt + X	Thoát khỏi chương trình.

hợp		

1.6. CHƯƠNG TRÌNH MẪU

1.7. Cấu trúc cơ bản:

Chương trình Pascal đơn giản nhất phải có hai từ khóa 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 khóa Begin và End.

Trong chương trình có thể có nhiều khối lệnh, tức có thể có nhiều càp từ khóa Begin va End.

1.8. 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.
```

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 hoàc 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 hoàc 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, thưviện SYSTEM màc định được chuyển vào chương trình mà ta không cần phải khai báo. Ví dụ một cách khai báo thư viên:

```
...
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 sau. Ví dụ một cách khai báo nhãn:

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

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

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

- Type: từ khóa dùng để khai bao các kiểu hàng dữ liệu sử dụng cho chương trình. Với từ khóa 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 sau). Một ví dụ về cách khai báo biến:

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

Ghi chú:

- Thứ tự các khai báo trên là điều bàt buộc, ta phải nàm 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 càp dấu {...} hoàc (*...*) lồng các câu chú thích vào bên trong nó.

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

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

Ví dụ 1:

```
Program  GioiThieu;
Begin
  Writeln ( ‘      Truong Cao dang Nghe Da Nang’);
  Write    ( ‘              99 Tô Hiến Thành          ‘ );
End.
```

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

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.
```

Ví dụ 3:

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

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

1.10. BÀI TẬP

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

```
Uses Crt;
```

```
Begin

Writeln('**********************************************************
**');
        Writeln('*  TRUONG CAO DANG NGHE DA NANG  *');
        Writeln('*                Xin chao ban !                  *');

Writeln('**********************************************************
**');
        Readln;
End.
```

3. Dịch và chạy chương trình trên.
4. Lưu chương trình vào đĩa với tên BAI1.PAS.
5. Thoát khỏi Pascal.
6. Khởi động lại Turbo Pascal.
7. Mở file BAI1.PAS.
8. Chèn thêm vào dòng: **CLRSCR;** vào sau dòng **BEGIN**
9. Dịch và chạy thử chương trình.
10. Lưu chương trình vào đĩa.
11. Thoát khỏi Pascal.

CHƯƠNG 2:

Tên chương : Các thành phần cơ bản
Mã chương: MH15-02
Mục tiêu:

- Trình bày và sử dụng được hệ thống kí hiệu và từ khóa.
- Mô tả được các kiểu dữ liệu.
- Trình bày được và vận dụng được các loại biến, hằng biểu thức cho từng chương trình cụ thể;
- So sánh được các lệnh, khối lệnh;
- Thực hiện được việc chạy chương trình.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung :

2.1. Hệ thống từ khóa và kí hiệu được dùng trong ngôn ngữ lập trình

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,...

2.2. Các kiểu dữ liệu cơ bản.

2.3. Cac kiểu dữ liệu dạng sô´

2.4. Kiểu số nguyên: (integer)

Một giá trị kiểu số nguyên là một phần tử của tập số nguyên mà có thể biểu diễn trên máy, nghĩa là nó là một tập nhỏ của các số nguyên chứ không phải là tất cả mọi số nguyên. Kiểu số nguyên được định nghĩa với các từ khóa sau:

TỪ KHÓA	PHẠM VI
INTEGER	-32768..32767
BYTE	0..255

WORD	0..65535
SHORTINT	-128..127
LONGINT	-2147483648..2147483647

2.5. Kiểu số thực: (Real)

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

TỪ KHÓA	PHẠM VI
REAL	$2.9x10^{-39}..1.7x10^{38}$
SINGLE	1.5E-39..1.7E+38
DOUBLE	5.0E-324..1.7E+308
EXTENDED	1.9E-4951..1.1E+4932

Lưu ý:

+ *Dạng viết thập phân bình thường như*:

  3.14 3.0 -24.567 -0.0089

+ *Dạng viết có phần mũ như*:

Gồm 2 phần, phần định trị và phần mũ viết sau chữ E để biểu diễn cơ số 10.

$$5678.12345 = 5.67812345*10^3$$

Sẽ được viết lại cho máy tính là:

5.67812345E+03

Phần định trị Phần mũ

2.6. Kiểu char, logic, string

2.1.1.1. Kiểu Char:

Dùng để biểu diễn các giá trị là các ký tư 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 đánh mã số từ 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 bang mã ASCII,.v.v.

Để hiển thị bảng mã ASCII, 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.
```

2.7. Kiểu Logic

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

Ví dụ:

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

2.8. 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.

2.9. Hằng, biến, hàm, các phép toán và biểu thức

2.10. Hằng:

Khái niệm : là đại lượng có giá trị không đổi trong suốt chương trình. Có các loại hằng số (nguyên và thực), hằng ký tự, hằng Boolean.

Trong chương trình, hằng được khai báo bằng cách đặt tên các hằng vào phần khai báo hằng sau từ khóa Const như sau:

```
CONST
TÊN_HẰNG= GIÁ_TRỊ_CỦA_HẰNG;
```

***Ví dụ*:**

```
Const
A = 5;
B = True;
C = 'S';
```

(*Ta nên sử dụng tên hằng dưới dạng gợi nhớ nhằm rõ ràng khi cần sửa*)

- 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).

2.11. Biến (Variable)

- 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 truc ghi nhơ dữ liệu vì vậy phải được quy định theo một kiểu dữ liệu nào đó, ví dụ kiểu Integer, Byte, Char,...

```
VAR
TÊN_BIẾN : KIỂU_DỮ_LIỆU_CỬA_BIẾN;
```

Ví dụ:

```
Var
A       : Real;
B, C    : Integer;
TEN     : String [20];
X       : Boolean;
Chon    : Char;
```

2.12. Biểu thức

Biểu thức là một công thức tính toán bao gồm các phép toán, hằng, biến, hàm, các dấu ngoặc.

Ví dụ: 5 + A * SQRT(B) / SIN(X)

Khi tính giá trị của biểu thức, luôn tuân theo thứ tự ưu tiên như sau:

Dấu ngoặc (,)	Biểu thức trong dấu ngoặc được ưu tiên trước
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 mức ưu tiên

=, <>, <=, >=, >, <, IN	Các phép tính quan hệ cùng mức ưu tiên

Và luôn tuân theo qui tắc sau:

- Các phép toán nào có thứ tự ưu tiên cao hơn sẽ được tính trước
- Nếu các phép toán có cùng ưu tiên sẽ được tính từ trái sang phải
- Phần trong ngoặc sẽ được tính trước
- Kiểu của biểu thức là kiểu của kết qủa sau cùng

2.13. Các lệnh, khối lệnh

Sau phần khai bao dữ liêu là phần lênh của chương trình. Phần này xác định các công việc ma chương trình phải thực hiện xư ly các dữ liệu đa được khai báo. Câu lệnh đươc chia thành hai loại:

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

+ Lệnh WITH.

2.14. Khôi lệnh

Là một nhóm câu lệnh đơn giản được đặt giữa 2 chữ Begin và End sẽ tạo nên câu lệnh hợp thành hay lệnh ghép với mẫu viết như sau:

```
BEGIN
 CÂU LỆNH 1;
 CÂU LỆNH 2;
 ...
 CÂU LỆNH N;
END;
```

2.15. Lệnh gán

Mục đích của lệnh này nhằm gán cho biến một giá trị sao cho phù hợp với kiểu khai báo của biến trước đó

Cách gán như sau:

```
TÊN_BIẾN := BIỂU THỨC;
```

Ví dụ:

```
c: = 'A';
i: = (20 + 5) * 2 Mod 4;
x: = 0.5;
```

2.16. Lệnh nhập

CÁCH VIẾT	Ý NGHĨA
READ(X1,X2,...,XN);	Nhập dữ liệu từ bàn phím vào các biến X1,X2,...Xn.
READLN(X1,X2,...,XN);	Nhập dữ liệu từ bàn phím vào các biến X1,X2,...Xn nhưng khi nhập xong, con trỏ xuống dòng.
READLN;	Dừng chương trình, đợi phím Enter để tiếp tục.
ASSIGN(F,TÊN_FILE);	Mở File F trên đĩa có tên là Tên_File
RESET(F);	Chuẩn bị đọc.
READ(F,X1,X2,...,XN);	Đọc các giá trị có ghi tên File F ra các biến X1,X2,...,Xn.
CH:=READ KEY;	Đọc một ký tự từ bàn phím vào biến ký tự CH
KEYPRESSED	Một hàm có giá trị là TRUE nếu có một phím được bấm và FALSE nếu ngược lại.
Chú ý: Biến F cần khai báo trước ở phần trên bởi lệnh: VAR F : TEXT;	

Ví dụ:

```
Program Vidu1;
Var
      i,j : Integer;
Begin
      Read(i,j);
      Writeln(i,j);
End.
```

- ✓ **Chú ý:**

- Các biến trong thủ tục Readln phải thuộc kiểu nguyên, thực, ký tự hoặc chuỗi 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.

- ✓ **Câu hỏi kiểm tra:**

Câu hỏi 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> thì kết quả thế nào?
=>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.
Câu hỏi 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 giá trị của 3 biến s1, s2, s3 là gì?
Đáp án: cả 3 biến s1, s2, s3 đều là chuỗi rỗng.

2.17. Lệnh xuất

CÁCH VIẾT	Ý NGHĨA
WRITE(X1,X2,..,XN);	Viết giá trị của các biến X1, X2, ... ra màn hình.
WRITELN(X1, X2,...,XN);	Tác dụng như trên nhưng khi viết xong có xuống dòng.
WRITELN;	Xuống một dòng.
WRITELN(I:N);	Viết giá trị của biến nguyên I vào n chỗ tính từ bên phải sang bên trái. (dạng viết có quy cách)
WRITELN(R:N:M);	Viết giá trị của biến thực R vào n chỗ và lấy m số lẻ thập phân. (dạng viết có quy cách)
WRITELN('ABC..F');	Viết ra nguyên văn chuỗi ABC..F
WRITELN(LST,X1,...,XN);	Viết gía trị của các biến ra máy in. Trước đó, trong chương trình phải có khai báo: USES PRINTER;
ASSIGN(F,TÊN_FILE);	Mở File F có tên là Tên_File
REWRITE(F);	Chuẩn bị viết.
WRITE(F,X1,X2,...,XN);	Ghi các giá trị của các biến X1,X2,...Xn vào File F.
CLOSE(F);	Đóng File F.
Chú ý: Biến F cần khai báo trước ở phần trên bởi lệnh: VAR F : TEXT;	

Ví dụ 1: Viết chương trình sau và xem kết quả trên màn hình:

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

➔Kết quả sau khi chạy chương trình trên:

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

Ví dụ 2 (về các dạng viết không quy cách): Viết chương trình sau và xem kết quả thực hiện các lệnh xuất trên màn hình. Từ đó rút ra nhận xét gì?

```
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.
```

Ví dụ 3 (Ví dụ về các dạng viết có quy cách): Viết chương trình sau và xem kết quả thực hiện các lệnh xuất trên màn hình. Từ đó rút ra nhận xét gì?

```
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.
```

2.18. Thực thi chương trình, nhập dữ liệu, nhận kết quả

Ví dụ: Viết chương trình tính diện tích S của hình thang với đáy dài a, đáy ngắn b, chiều cao h 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.
```

Thực thi chương trình: Ctrl+F9

Kết quả chương trình:

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

2.19. BÀI TẬP

Câu 1. Viết chương trình Pascal để in ra màn hình những câu ca dao sau:

Trong đầm gì đẹp bằng sen
Lá xanh bông trắng lại chen nhị vàng
Nhị vàng bông trắng lá xanh
Gần bùn mà chẳng hôi tanh mùi bùn

Câu 2. Viết chương trình tính cạnh huyền của tam giác vuông có 2 cạnh là a và b theo công thức pitago : $c^2 = a^2 + b^2$

Câu 3. Viết chương trình của một tam giác khi biết 3 cạnh a, b,c
Biết công thức tính diện tích tam giác : $S=\sqrt{p*(p-a)*(p-b)*(p-c)}$

$$p=\frac{1}{2}*(a+b+c)$$

Câu 4. Viết chương trình tính giá trị của biểu thức:

$$F(x) \quad \frac{5x^3 - \sqrt{x^4 - 1} - |x - 5|}{\sin(x^2) - arctg(|x - 1|)}$$

Câu 5. Viết chương trình nhập vào số giờ, in ra màn hình số phút tương ứng. Ví dụ: 5 giờ = 300 phút.

Câu 6. Viết chương trình nhập vào số met, in ra màn hình số dm, cm, mm tương ứng. Ví dụ: 5m=50dm=500cm=5000mm.

Câu 7. Viết chương trình tính chu kỳ con lắc đơn $T=2\Pi*\sqrt{\frac{l}{g}}$, với g là gia tốc trọng trường 9,18m/s^2.

CHƯƠNG 3:

Tên chương: Các cấu trúc điều khiển
Mã chương/ bài:MH15-03
Mục tiêu :

- Trình bày được lệnh có cấu trúc;
- Vận dụng được các lệnh cấu trúc: cấu trúc lựa chọn, cấu trúc lặp xác định và lặp vô định;
- Vận dụng được các lệnh bẻ vòng lặp.
- Thực hiện các thao tác an toàn với máy tính.

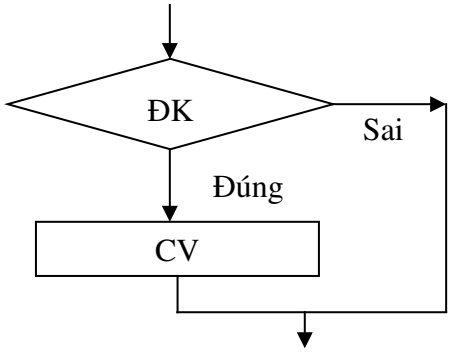
Nội dung :

3.1. Các lệnh cấu trúc lựa chọn

3.2. Lệnh IF

3.3. Dạng không đầy đủ

Cú pháp : IF <biểu thức điều kiện> THEN <công việc>;
Lược đồ cú pháp :



Giải thích: Khi gặp trường hợp này máy kiểm tra <biểu thức điều kiện>, nếu biểu thức này có giá trị TRUE (tức là đúng như điều kiện đặt ra) thì máy thực hiện <công việc >, các lệnh liền sau <công việc> không phụ thuộc vào biểu thức điều kiện.
Ví dụ : Viết chương trình nhập từ bàn phím 2 số nguyên a, b. Kiểm tra và cho biết số nào lớn hơn.

```
 Var  a, b,max : Integer;
Begin
 Write( ' Nhap so a: ' );
 Readln(a);
 Write( ' Nhap so b: ' );
```

```
 Readln(b);
 max:=a;
 If max < b then max:=b
  Write( ' So lon hon la ', max)
 Readln;
End.
```

3.4. Dạng đầy đủ

Cú pháp :

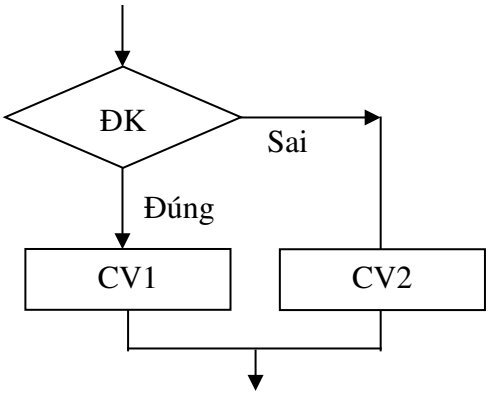
IF <biểu thức điều kiện> THEN

<công việc 1 >

ELSE

<công việc 2>;

Lưu đồ cú pháp :



Giải thích : Khi gặp trường hợp này máy kiểm tra <biểu thức điều kiện>, nếu biểu thức này có giá trị TRUE (tức là đúng như điều kiện đặt ra) thì máy thực hiện <công việc 1> nếu ngược lại, tức <biểu thức điều kiện> có giá trị FALSE thì máy thực hiện <công việc 2> . Các lệnh liền sau <công việc 2> không phụ thuộc vào biểu thức điều kiện.

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 khóa END trước ELSE không được đặt dấu ';'.

Ví dụ : Viết 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;
End.
```

3.5. Lệnh CASE..OF

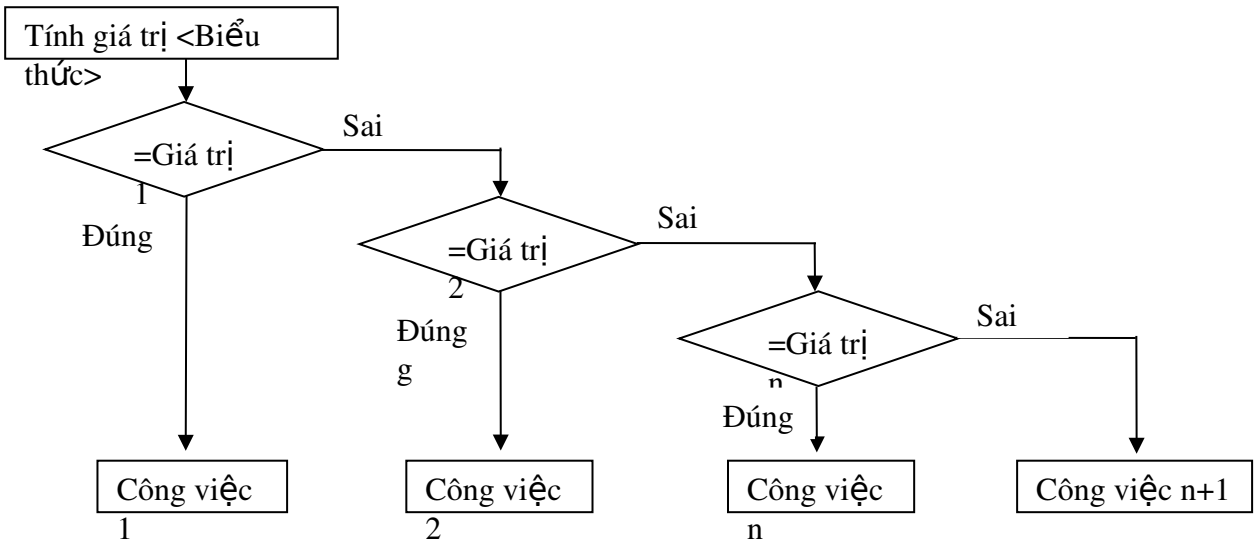
Câu lệnh IF ở trên chỉ rẽ vào một trong hai nhánh tương ứng với giá trị của biểu thức điều kiện. Còn lệnh CASE (rẽ nhánh theo giá trị) cho phép lựa chọn để thực hiện một trong nhiều công việc tùy theo giá trị của biểu thức.

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;

- ✓ **Lưu ý : Lệnh CASE có thể không có phần [ELSE <lệnh n +1>;]**

Lưu đồ cú pháp :



Giải thích :

- Tập_hằng_i (i = 1,..., n) có thể bao gồm các hằng và các miền giá trị (của hằng), ví dụ:

```
 3 :              <lệnh 1>;
 5, 10.. 15 :     <lệnh 2>;
 'A', Chr(152) :  <lệnh 3>;
```

‘0’.. ’9’ : <lệnh 4>;

- 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).

- 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.

- 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.

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.
```

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

- Nhập tháng vào biến Thang, nhập năm vào biến Nam
- 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:
    - Trường hợp Nam chia hết cho 4: SoNgay := 29;
    - Trường hợp Nam không chia hết cho 4: SoNgay := 28;
  + Tháng 4, 6, 9, 11: SoNgay := 30;
- In nôi dung biến SoNgay.

(Sinh viên tự viết chương trình)

3.6. Cấu trúc vòng 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 lênh 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.

3.7. Vòng lặp FOR:

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

3.8. 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 kiểu vô hướng đếm được (như nguyên, logic, ký tự, liệt kê).

Giải thích:

(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.

3.9. Dạng lùi:

Cú pháp:

FOR <Biến := Biểu_thức1> DOWNTO <Biểu_thức2> DO <Lệnh>

Giải thích:

(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).

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 đôi. 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.

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  3  do
   Writeln( i , ' => ', ' Chao cac ban ' );
  Readln;
End;
```

Ví du 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.
```

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) );
```

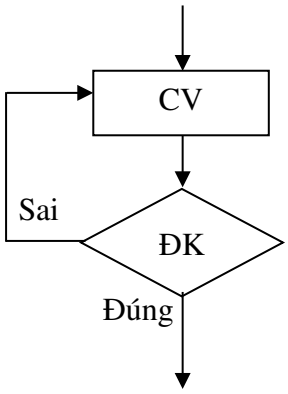
```
    If  (i+1)  mod 22 =  0  then
      Begin
      Write( ‘ An phim bat ky de xem tiep ! ‘ );
      Readln;
       End;
   End;
Readln;
End.
```

3.10. Câu lệnh Repeat:

Cú pháp:

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

Lưu đồ cú pháp :



Giải thích:

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.

Chú ý:

- Các lệnh nằm giữa REPEAT và UNTIL không có từ khóa 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 may.

Ví dụ1: Chương trình yêu cầu nhập vào một mật khẩu là 'danavtc' thì mới thoát khỏi chương trình.

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

Ví dụ 2 : Viết chương trình nhập vào n và in ra kết quả n!.

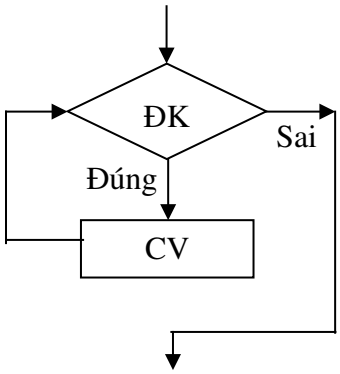
```
uses crt;
var n,i,p:integer;
Begin
write('n=');read(n);
i:=1;
p:=1;
repeat
p:=p*i;
i:=i+1;
until (i>n);
write(n,'! =',p);
readln;
End.
```

3.11. Câu lệnh While:

Cú pháp:

WHILE < Biểu thức điều kiện > DO < công việc >;

Lưu đồ cú pháp :



Giải thích: Gặp lệnh này trước tiên máy kiểm tra < Biểu thức điều kiện >, nếu nó có giá trị TRUE thì thực hiện < Lệnh > và sau đó quay lại kiểm tra < Biểu thức điều kiện> và quá trình cứ tiếp tục như vậy. Nếu < Biểu thức điều kiện > 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.

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.

Ví dụ 1 : Viết chương trình nhập vào n và in ra kết quả n!.

```
uses crt;
var n,i,p:integer;
Begin
write('n=');read(n);
i:=1;
p:=1;
while (i<=n) do
        begin
        p:=p*i;
        i:=i+1;
        end;
write(n,'! =',p);
readln;
End.
```

Ví dụ2: 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.
```

3.12. Các lệnh chuyển điều khiển:

3.13. Lệnh Goto:

Cú pháp: GOTO <tên nhãn>;

Trong đó, 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 chương 1. Khi gặp lệnh Goto <tên nhãn>, máy nhảy không điều kiện đến thực hiện câu lệnh sau nhãn.

Lệnh Goto chỉ cho pháp nhảy từ vị trí này đến vị trí khác trong cùng một thân hàm, thủ tục, cho pháp 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.

3.14. 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.

Ví dụ: In ra màn hình 3 dãy số từ 1 đến 10.

```
uses crt;
var i,j:integer;
begin
clrscr;
for i:=1 to 3 do
   begin
   writeln;
   for j:=1 to 30 do
   begin
   if (j=11) then
   break;
   write(j,' ');
```

```
   end;
   end;
readln;
END.
```

3.15. 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.
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.
```

3.16. Kết hợp các cấu trúc điều khiển trong chương trình .

Ví dụ 1 : In ra 5 dãy số từ 1 đến 9

```
Uses  CRT;
Var  i, j : Integer;
Begin
  ClrScr;
  for  j := 1  to  5  do
         Begin
         Writeln;
         Writeln( ' j = ' , j );
          for  i := 1  to  9  do
         Write( i, '  ' );
Readln;
End;
```

```
Readln;
End.
```
Ví dụ 2 : In ra màn hình tam giác vuông như sau :

```
*
**
***
****
*****
Uses crt;
Var i,j,h: integer;
Begin
        clrscr;
        //khai bao bien
        writeln('chuong trinh ve hinh');
        write('nhap chieu cao cua hinh h= ');readln(h);
        //Ve hinh
        i:=1;
        while (i<=h) do
        Begin
                write(" ");
                j:=1;
                while (j<=i) do
                Begin
                        write('*');
                        j:=j+1;
                end;
                writeln;
                i:=i+1;
        end;
        readln;
End.
```

3.17. Bài tập

3.18. Câu lệnh IF

Bài 1: Viết chương trình tìm số lớn nhất trong 2 số nhập từ bàn phím. (Viết dưới dạng **if** thiếu và **if** đủ).

Bài 2: Viết chương trình tìm số lớn nhất, nhỏ nhất trong 4 số nguyên cho từ bàn phím

Bài 3: Nhập tuổi của một người, hiển thị ra màn hình:

- Tuổi < 18: Trẻ vị thành niên

- 18 ≤ tuổi ≤ 60: Người lớn
- Tuổi > 60: Người già

Bài 4: Viết chương trình kiểm tra xem một năm nào đó có nhuận hay không

Bài 5: -Viết chương trình giải phương trình bậc nhất: ax+b=0

-Viết chương trinhg giải phương trình bậc hai: $ax^2+bx+c=0$

Hướng dẫn :

Bài 1:

❖Thuật toán:

- Khai báo biến a, b, max
- Nhập hai số a, b từ bàn phím
- Dùng câu lệnh if:
  - if thiếu:
    - Nếu a<b thì max = b
    - Nếu a>b thì max = a
  - if đủ:
    - Nếu a<b thì max = b
    - Ngược lại, max = a
- Xuất giá trị max tìm được ra màn hình

Bài 2:

❖Thuật toán:

- Khai báo biến a, b, c, d, max
- Nhập bốn số a, b, c, d từ bàn phím
- Gán: max=a (giả sử giá trị lớn nhất ban đầu là a)
- Dùng câu lệnh if để tìm giá trị lớn nhất:
  - Nếu max<b thì max = b
  - Nếu max<c thì max = c

  Thuật toán này được gọi là thuật toán lính canh
- Xuất giá trị max tìm được ra màn hình

Bài 3:

❖Thuật toán:

- Khai báo biến tuoi
- Nhập tuoi từ bàn phím
- Dùng câu lệnh if:

  O Nếu tuoi<18 thì in ra màn hình câu `ban la tre vi thanh nien`

  O Ngược lai, nếu tuoi<=60 in ra màn hình câu `ban la nguoi lon`

  Ngược lại, in ra câu `ban la nguoi gia`

Bài 4:

❖Thuật toán:

- Khai báo biến nam

- Nhập năm cần kiểm tra có phải là năm nhuận hay không
- Nếu (nam mod 4= 0) thì năm đó là năm nhuận
- Ngược lại, năm đó không phải là năm nhuận

Bài 5:

Giải phương trình bậc nhất: ax+b=0

❖Thuật toán:

- Khai báo biến a, b, x
- Nhập hệ số a, b của phương trình bậc nhất từ bàn phím
- Cách giải phương trình bậc nhất:
  - Nếu a=0
    - Nếu b=0 thì phương trình vô số nghiệm
    - Ngược lại, phương trình vô nghiệm
  - Ngược lại, phương trình có nghiệm x = -b/a

3.19. Câu lệnh rẽ nhánh có điều kiện switch..case

Bài 1: Lập chương trình đọc từ bàn phím một số nguyên n (1≤n≤10) rồi đưa ra tiếng Anh của số đó. Chẳng hạn, nếu gõ vào n = 4 thì in ra Four

Bài 2: Nhập vào một tháng bất kỳ trong năm và cho biết tháng đó thuộc mùa nào (xuân, hạ, thu, đông) trong năm.

Bài 3: Viết chương trình đổi một năm dương lịch sang năm âm lịch. Biết rằng:

Can = (giáp, ất, bính, đinh, mậu, kỹ, canh, tân, nhâm, qui)

Chi = (tí, sửu, dần, mão, thìn, tị, ngọ, mùi, thân, dậu, tuất, hợi).

Hướng dẫn :

Bài 1**:**

❖Thuật toán:

- Khai báo biến: so, kq
- Nhập một số bất kỳ sao cho 1<= so <=10 từ bàn phím
- Cách chuyển số vừa nhập sang tiếng anh (dùng câu lệnh case..of):

```
Case (so) of
        1: kq=”One”; break;
        2: kq=”Two”; break;
        3: kq=”Three”; break;
        4: kq=”Four”; break;
        5: kq=”Five”; break;
        6: kq=”Six”; break;
        7: kq=”Seven”; break;
        8: kq=”Eight”; break;
        9: kq=”Nine”; break;
```

10: kq=”Ten”; break;
End;

- Xuất kq ra màn hình

Bài 2**:**

❖Thuật toán:

- Khai báo biến: thang, mua
- Nhập một tháng bất kỳ từ bàn phím
- Kiểm tra tháng đó thuộc mùa nào trong năm (dùng câu lệnh case..of):

Case (thang) of
1..3: mua=”mua xuan”; break;
4..6: mua=”mua ha”; break;
7..9: mua=”mua thu”; break;
10..12: mua=”mua dong”; break;
End;

- Xuất muà ra màn hình

Bài 3:

❖Thuật toán:

- Khai báo biến: nam, can, chi
- Nhập một năm bất kỳ từ bàn phím
- Thực hiện chuyển năm âm lịch sang năm dương lịch (dùng câu lệnh case..of):

Case (nam mod 10) of
1: can = ”giap”; break;
2: can = ”at”; break;
3: can = ”binh”; break;
4: can = ”dinh”; break;
5: can = ”mau”; break;
6: can = ”ky”; break;
7: can = ”canh”; break;
8: can = ”tan”; break;
9: can = ”nham”; break;
10: can = ”qui”; break;
End;
Case (nam mod 12) of
1: chi = ”ti”; break;
2: chi = ”suu”; break;
3: chi = ”dan”; break;
4: chi = ”mao”; break;
5: chi = ”thin”; break;
6: chi = ”ti”; break;
7: chi = ”ngo”; break;
8: chi = ”mui”; break;

```
            9: chi = ”than”; break;
            10: chi = ”dau”; break;
            11: chi = ”tuat”; break;
            12: chi = ”hoi”; break;
        End;
```

- Xuất giá trị can, chi ra màn hình

3.20. Câu lệnh for

Bài 1: Hãy làm theo yêu cầu sau:

a.Viết chương trình kiểm tra **n** có phải là **số nguyên tố** hay không, với số n được nhập vào từ bàn phím

b.Viết chương trình nhập vào số n và in ra các số nguyên tố có từ 1-> n, đếm có bao nhiêu số nguyên tố như vậy.

Bài 2: Tính các tổng sau:

a. S=1+2+3+…+n
b. S= $1^2+2^2+3^2+\ldots+n^2$
c. S=1/1+1/2+1/3+…+1/n

Bài 3: Viết chương trình in ra các số từ 1 đến 100 theo dạng sau:

```
1    2... .............10
11   12   13.............20
.....................................
91  92   93..... 100
```

Hướng dẫn :

Bài 1a:

❖Thuật toán:

d. Khai báo biến n, i
e. Nhập số n
f. Thuật toán kiểm tra n có phải là số nguyên tố không:
    a. Cho i chạy từ 2 đến n
        i. Nếu n mod i = 0 thì thoát;
    b. Nếu i = n thì in ra màn hình n là số nguyên tố
    c. Ngược lại, n không phải là số nguyên tố

Bài 1b:

❖Thuật toán:

g. Khai báo biến n, i, j
h. Nhập số n
i. Tìm các số nguyên tố có được từ 1->n:
    a. Khởi tạo biến đếm count=0
    b. Cho i chạy từ 2 đến n

i. Cho j chạy từ 2 đến i
    1. Nếu i mod j = 0 thì thoát
ii. Nếu j=i thì:
    1. i là số nguyên tố
    2. count=count+1;

j. In giá trị count và các số nguyên tố tìm được ra màn hình

Bài 2a:

❖Thuật toán:

- Khai báo biến S, i, n
- Nhập n từ bàn phím
- Tính tổng S:

```
S=0
Cho i chạy từ 1 đến n
    S=S+i
```

- Xuất S ra màn hình

Bài 2b:

❖Thuật toán:

- Khai báo biến S, i, n
- Nhập n từ bàn phím
- Tính tổng S:

```
S=0
Cho i chạy từ 1 đến n
    S=S+1/i
```

- Xuất S ra màn hình

Bài 6:

❖Thuật toán:

- Khai báo biến i
- Tạo bảng số:

```
Cho i chạy từ 1 đến 100 làm
    Nếu i mod 10= 1 thì in xuống dòng và i ra màn hình
    Ngược lại, in giá trị i ra màn hình
```

3.21. Cấu trúc vòng lặp while

Bài 1: Viết chương trình tính tổng 100 số nguyên dương đầu tiên

Bài 2: Viết chương trình tính tổng sau:

- S=1+3+5+…+2n-1
- S=2+4+6+…+2n
- $S=1/1^2+1/2^2+1/3^2+\ldots+1/n^2$

Bài 3: Viết chương trình nhập vào một số nguyên n, và tính tổng các chữ số của n

Ví dụ: 156, tổng các chữ số là 1+5+6=12

Bài 4: Viết chương trình in ra màn hình như sau:

*	ooooo	x
**	oooo	xxx
***	ooo	xxxxx
****	oo	xxxxxxx
*****	o	xxxxxxxxx

Bài 5: Viết chương trình in ra dãy **fibonaxi** 1 1 2 3 5 8 13 21 34 … nhỏ hơn giá trị N nhập vào từ bàn phím

$F_0=F_1=1$

$F_n=F_{n-1}+F_{n-2}$

Hướng dẫn :

Bài 1:

❖Thuật toán:

- Khai báo biến i, S
- Tính tổng 100 số nguyên dương đầu tiên
  - Gán: S=0; i=1
  - Chừng nào i<=100 làm
    - S=S+i
    - i=i+1
- Xuất giá trị S ra màn hình

Bài 2a:

❖Thuật toán:

- Khai báo biến i, S, n
- Tính tổng S=1+3+5+..+2n-1
  - Gán: S=0; i=1
  - Nhập n từ bàn phím
  - Chừng nào i<=2*n-1 làm
    - S=S+i
    - i=i+2
- Xuất giá trị S ra màn hình

Bài 2b:

❖Thuật toán:

- Khai báo biến i, S, n
- Tính tổng S=2+4+6+..+2n
  - Gán: S=0; i=2
  - Nhập n từ bàn phím
  - Chừng nào i<=2*n làm
    - S=S+i
    - i=i+2
- Xuất giá trị S ra màn hình

Bài 3:

❖Thuật toán:

- Khai báo biến t, S, n
- Tính tổng các chữ số của số n nhập vào từ bàn phím
    - Nhập n từ bàn phím
    - Gán S=0
    - Chừng nào n>0 làm
        - t= mod 10
        - n=n div 10
        - S=S+t
- Xuất giá trị S ra màn hình

Bài 4a:

❖Thuật toán:

- Khai báo biến i, j, h
- Vẽ hình a:
    - Nhập h từ bàn phím
    - Gán i=1
    - Chừng nào i<=h làm
        - j=1
        - Chừng nào j<=i làm

            Xuất ký tự * ra màn hình

            Tăng j lên 1 đơn vị: j=j+1
        - In ký tự xuống dòng
        - Tăng i lên 1 đơn vị: i=i+1

Bài 4b:

❖Thuật toán:

- Khai báo biến i, j, h
- Vẽ hình b:
    - Nhập h từ bàn phím
    - Gán i=1
    - Chừng nào i<=h làm
        - j=h
        - Chừng nào j=>i làm

            In ký tự * ra màn hình

            Giảm j xuống 1 đơn vị: j=j-1
        - In ký tự xuống dòng
        - Tăng i lên 1 đơn vị: i=i+1

Bài 4c:

❖Thuật toán:

- Khai báo biến i, j, h
- Vẽ hình b:
    - Nhập h từ bàn phím

- Gán i=1
- Chừng nào i<=h làm
  - j=h-1
  - Chừng nào j=>i làm
    In ký tự trống ra màn hình
    Giảm j xuống 1 đơn vị: j=j-1
  - Gán j=1
  - Chừng nào j<=2*i-1 làm
    In ký tự * ra màn hình
    Tăng j lên 1 đơn vị: j=j+1
  - In ký tự xuống dòng
  - Tăng i lên 1 đơn vị: i=i+1

Bài 5:

❖Thuật toán:

- Khai báo biến n, f1, f2, f
- Nhập n từ bàn phím
- Gán f1=1, f2=1, f=2
- In f1, f2 ra màn hình
- Chừng nào (f1+f2<=n) làm
  - f=f1+f2
  - f1=f2
  - f2=f
  - in f ra màn hình

3.22. Cấu trúc vòng lặp repeat..until

Bài 1**:** Hãy đánh vào và chạy chương trình sau để xem và giải thích kết quả:

```
main()
{
     int i ;
     for (i=1; i<=100 ; i++)    printf("%5d ", i );
     getch();
}
```

Sau đó hãy thay lệnh lặp ***for*** bằng lệnh lặp ***do..while***, kết quả có gì thay đổi không?

Bài 2: Nhập vào n, in ra màn hình kết quả n!

Bài 3: Viết chương trình nhập vào số nguyên dương n, hãy tính tổng và tích các chữ số của nó

Bài 4: - **Số hoàn hảo** là số bằng tổng các ước số của nó. Ví dụ: 6 là số hoàn hảo vì, ước số của 6: 1, 2, 3 và tổng các ước số của nó bằng 6 (1+2+3=6)

- Nhập vào số n, kiểm tra n có phải là số hoàn hảo hay không

- Nhập vào số n, hãy xuất ra màn hình các **số hoàn hảo** từ 1->n, đếm có bao nhiêu số hoàn hảo

Hướng dẫn :

Bài 2:

❖Thuật toán:

- Khai báo biến n, i, gth
- Nhập n từ bàn phím
- Gán i=2, gth=1
- Làm
    - gth=gth*i
    - i=i+1

  Chừng nào i<=n
- Xuất gth ra màn hình

Bài 4a:

❖Thuật toán:

- Khai báo biến n, i, S
- Nhập n từ bàn phím
- Gán S=0, i=1
- Làm
    - Nếu n mod i= = 0 thì S=S+i
    - i=i+1

  Chừng nào i<=n
- Nếu S = = n thì n là số hoàn hảo
- Ngược lại, n không phải là số hoàn hảo
- Xuất n ra màn hình

CHƯƠNG 4:

Tên chương : Hàm và thủ tục
Mã chương: MH15.04.
Mục tiêu :

- Trình bày được khái niệm hàm, thủ tục;
- Trình bày được qui tắc xây dựng hàm, thủ tục và vận dụng được khi thiết kế xây dựng chương trình;
- Phân biệt được cách sử dụng tham số, tham biến;
- Sử dụng được các lệnh kết thúc và lấy giá trị trả về của hàm.
- Thực hiện các thao tác an toàn với máy tính.

4.1. Khái niệm chương trình con

Trong chương trình, có những đoạn cần phải lập đi, lập lại nhiều lần ở những chỗ khác nhau. Để tránh phải viết lại các đoạn đó người ta thường phân chương trình ra thành nhiều module, mỗi module giải quyết một công việc nào đó, các module như vậy là những chương trình con (subprogram).

Một tiện lợi khác của việc sử dụng module là ta có thể dễ dàng kiểm tra tính đúng đắn của nó trước khi ráp nối vào chương trình chính. Do đó việc xác định sai sót và tiến hành điều chỉnh trong chương trình sẽ thuận lợi hơn.

Trong Pascal chương trình con được viết dưới dạng hàm (FUNCTION) hoặc thủ tục (PROCEDURE). Hàm và thủ tục đều là những chương trình con, nhưng hàm khác thủ tục ở chỗ hàm trả về một giá trị cho lệnh gọi thông qua tên hàm còn thủ tục thì không. Do đó ta chỉ dùng hàm khi thoả mãn các yêu cầu sau.

- Ta muốn nhận một kết quả và chỉ một mà thôi.
- Ta cần dùng tên chương trình con (chứa kết quả đó) để viết trong các biểu thức.

Nếu không thỏa hai yêu cầu trên thì ta dùng thủ tục.

4.2. Cấu trúc chương trình có sử dụng chương trình con

Trong một chương trình các chương trình con được bố trí ngay sau phần khai báo biến. Cấu trúc tổng quát một chương trình trong ngôn ngữ Pascal như sau:
PROGRAM tên_chương_trình;

```
USES tên các Unit (*Khai báo các đơn vị chương trình cần thiết*)
LABEL          (*Khai báo nhãn*)
CONST          (*Khai báo hằng*)
TYPE           (*Định nghĩa kiểu dữ liệu mới*)
VAR            (*Khai báo biến*)
PROCEDURE Tên_CTC1 (danh sách các tham số hình thức);
Begin
……..   (*thân thủ tục thứ nhất*)
End;

PROCEDURE Tên_CTC2 (danh sách các tham số hình thức);
Begin
……..   (*thân thủ tục thứ hai*)
End;

FUNCTION Tên_Hàm1(danh sách các tham số hình thức);
Begin
……..   (*thân hàm thứ nhất*)
End;
FUNCTION Tên_Hàm2(danh sách các tham số hình thức);
Begin
……..   (*thân hàm thứ nhất*)
End;

……..
BEGIN
……..   (*chương trình chính*)
END.
```

4.3. Các hàm và thủ tục trong ngôn ngữ lập trình

4.4. Haøm

4.5. Cấu trúc một hàm

Hàm là một chương trình con tính toán trả về cho ta một giá trị kiểu vô hướng. Cấu trúc hàm như sau:

FUNCTION <Tên hàm>[(<Th.số>:<Kiểu>[;<Th.số>: <Kiểu>])]: <KiểuKQ>;	(Header)
[VAR <Biến>:<Kiểu>[;<Biến>: <Kiểu>]]	Khai báo các biến cục bộ nếu có.

BEGIN **<các câu lệnh>** END**;**	Thân hàm

- Tên hàm là một danh biểu, phải tuân thủ theo qui tắc đặt danh biểu đã đề cập ở chương I.

- Một hàm có thể không có hoặc có một hoặc nhiều tham số. Trong trường hợp có nhiều tham số có cùng một kiểu dữ liệu thì ta có thể viết chúng cách nhau bởi dấu **,** (phẩy). Ngược lại, các tham số hình thức khác kiểu nhau thì phải cách nhau dấu **;** (chấm phẩy).

- KiểuKQ là một kiểu vô hướng, nó phản ảnh kiểu của giá trị mà hàm trả về lại sau khi chạy xong. Ví dụ, ta khai báo hàm như sau:

FUNCTION TEST(x,y:Integer; z:Real): Real;

Đây là một hàm có tên là TEST, với 3 tham số, x và y thuộc kiểu Integer, z thuộc kiểu real, hàm trả về một kết quả kiểu real.

- Trong hàm, ta có thể sử dụng các hằng, kiểu, biến dùng riêng trong nội bộ hàm.

- Thông thường mục đích sử dụng hàm là để lấy trị trả về do đó cần lưu ý gán kết quả cho tên hàm trong thân hàm.

4.6. Phương pháp gọi hàm

Lời gọi hàm trong chương trình chính đươch thực hiện như sau:

Tên_hàm (danh sách các tham số thực)

4.7. Ví dụ

Ví dụ 1: Ta xây dựng hàm **DT** truyền tham số vào là bán kính của hình tròn, hàm này sẽ trả về diện tích của hình tròn đó.

```
Program TinhDienTich;
Uses Crt;
VAR  BanKinh: real; Ch: Char;
{-----------------------------------------}
Function DT(Radius:Real):Real;
Begin
      DT := PI * Radius* Radius;
End;
{-----------------------------------------}
```

Phép gán để trả về giá trị cho tên hàm.

```
Begin
        Clrscr;
        Repeat
                Write('Nhập bán kính: '); Readln(BanKinh);
                Writeln('Diện tích hinh tron tuong ung: ' ,DT(Bankinh):0:2);
                Writeln;
                Write('Tiếp tục (C/K)? ');
                        Repeat
                                ch:=readkey;
                        Until Upcase(ch) in ['C','K'];
        Until UpCase(Ch) = 'K';  {Lưu ý: 'K' in hoa}
End.
```

Ví dụ 2:

```
Program TinhGiaithua;
USES CRT;
Var Num:longint; Ch:char; X,Y:byte;
{-------------------------------------------}
Function GiaiThua(m: longint): longint;
Var Tam, Dem:Longint;
BEGIN
IF (M<0) THEN
  Begin
          Write('Khong tinh duoc'); HALT(1);
        End
ELSE
        Begin
      Tam:=1;
      For Dem:=1 to m do Tam:=Tam*Dem;
      GiaiThua:=Tam;
        End;
 END;
{-------------- Chương trình chính -------------------------}
BEGIN
        Writeln('CHUONG TRINH TINH GIAI THUA.');
REPEAT
                Write('Cho so nguyen muon tinh giai thua. M= ');
                X:=WhereX; Y:=WhereY;
                REPEAT
Gotoxy(X,Y); CLREOL; Readln(Num);
                        UNTIL (Num>=0);
                Writeln(M,'! = ',GiaiThua(Num));
```

```
            REPEAT
               Write('Tinh nua khong ? (C/K) :'); CH:=READKEY;
            UNTIL Upcase(Ch) in ['C','K'];
         Writeln(Ch);
UNTIL Upcase(Ch)='K';
Readln
END.
```

4.8. Thủ tục

4.9. Cấu trúc một thủ tục

Cấu trúc của một thủ tục như sau:

PROCEDURE <Tên>(<Th.số>:<Kiểu>[;<Th.số>: <Kiểu>]): <Kiểu>;	(Header)
[VAR <Biến>:<Kiểu>[;<Biến>: <Kiểu>]	Khai báo các biến cục bộ nếu có.
BEGIN **<các câu lệnh>** END**;**	Thân thủ tục.

Như vậy cấu trúc của một thủ tục cũng tương tự như cấu trúc của một hàm. Chỉ có hai điều khác:

- Header bắt đầu bằng từ khóa Procedure thay vì Function.
- Không có câu lệnh gán <Tenham:=GiaTri;> trong thân Procedure.

4.10. Phương pháp gọi thủ tục

Lời gọi thủ tục trong chương trình chính được thực hiện như sau:

Tên_thủ tục (danh sách các tham số thực)

4.11. Ví dụ:

Thủ tục INSO sau sẽ in các số từ 1 đến giá trị biến truyền vào. Với ***n*** là tham số thực tế, ***So*** là tham số hình thức.

```
Program TEST;
Var n: Integer;
{---------------------------------------}
Procedure INSO(So: Integer);
Var  i: Integer;
Begin
        For i := 1 to So do
                Write( i:10 );
End;
{------------ Chương trình chính --------------------}
Begin
        Write('Nhập một số bất kỳ lớn hơn không: '); Readln(n);
        INSO( n );
        Readln;
End.
```

4.12. Tham trị và tham biến

Các tham số của chương trình con chính là các dữ liệu cần thiết nhập vào để xử lý các phép toán trong chương trình con sử dụng. Các tham số này được gọi là tham số hình thức, bởi nó chỉ mang danh nghĩa là các đối số của chương trình con, chứ về mặt bản chất dữ liệu nó lại mang thông tin của các biến trong chương trình chính. Các tham số này có 2 loại: Tham biến và Tham trị. Các chương trình con có thể có nhiều loại Tham số hình thức khác nhau về kiểu tham số hay về kiểu dữ liệu của tham số.

4.13. Tham biến

Tham biến: Là loại tham số hình thức mà giá trị của nó có thể thay đổi được trong các phép xử lý tính toán của chương trình con. Có thể dữ liệu nạp vào chương trình con là A, nhưng sau khi ra khỏi chương trình con (kết quả sau khi thực hiện chương trình con) nó lại mang kết quả B. Tham biến là tham số hình thức được khai báo ở chương trình con và bắt buộc phải được khai báo với từ kháo khai báo VAR. Các chương trình con có thể có nhiều loại tham biến, và cách khai báo các tham biến giống hệt như bạn khai báo biến trong chương trình chính.

4.14. Tham trị

Tham trị: Là loại tham số hình thức mà giá trị của nó không thể thay đổi được trong các phép xử lý tính toán của chương trình con. Dữ liệu nạp vào chương trình con là A, nhưng sau khi ra khỏi chương trình con (kết quả sau khi thực hiện chương trình con) nó vẫn phải là A. Chính vì vậy, trong chương trình con bạn không thể nào thực hiện 1 phép toán làm thay đổi giá trị của

tham trị, nếu có máy sẽ báo lỗi.Tham trị là tham số hình thức được khai báo ở chương trình con và không bắt buộc phải được khai báo với từ kháo khai báo VAR. Các chương trình con có thể có nhiều loại tham trị, và cách khai báo các tham trị giống hệt như bạn khai báo biến trong chương trình chính.

4.15. Biến toàn cục và biến địa phương

4.16. Biến toàn cục

Biến khai báo ở đầu chương trình mẹ được gọi là biến toàn cục. Nó có tác dụng trong toàn bộ chương trình, kể cả các chương trình con. Khi thực hiện chương trình máy dành các ô nhớ ở vùng dữ liệu (data) để lưu trữ giá trị của biến.

Mở rộng ra tất cả các đối tượng (kể cả dữ liệu, hằng, biến, hàm, thủ tục) khai báo trong chương trình mẹ được gọi là đối tượng toàn cục. Như vậy một kiểu dữ liệu đã được định nghĩa trong một chương trình mẹ thì đương nhiên được phép sử dụng trong các chương trình con của nó.

4.17. Biến địa phương

Biến địa phương (hay còn gọi là biến cục bộ) là biến khai báo ở trong chương trình con và chỉ có tác dụng trong nội bộ chương trình con đó.

Biến địa phương có thể trùng tên với biến toàn cục song máy dành các ô nhớ khác trong vùng nhớ ngăn xếp (Stack) để lưu giữ các giá trị của biến. Kết thúc chương trình con máy sẽ giải phóng ô nhớ của biến địa phương dung vào việc khác, các giá trị lưu trữ trong biến này sẽ không còn.

Trường hợp, trong chương trình con lại có các chương trình con khác thì biến địa phương của chương trình con cấp trên lại được xem là biến toàn cục đối với chương trình con cấp dưới.

Một biến sau khi được khai báo trong một chương trình sẽ chỉ có tầm tác dụng trong bản thân chương trình đó và các chương trình con của nó. Biến này không có tác dụng trong các chương trình cùng cấp khác hoặc trong các chương trình con của chương trình khác. Điều này có nghĩa là các chương trình con và chương trình mẹ có thể có nhiều biến trùng tên, nhưng tầm tác dụng thì khác nhau do đó tính toàn vặn của dữ liệu luôn được bảo đảm.

4.18. Bài tập:

Nội dung

k. Cách thức xây dựng thủ tục trong chương trình
l. Việc gọi thủ tục thực hiện trong chương trình chính
m. Cách thức xây dựng hàm trong chương trình
n. Việc gọi hàm thực hiện trong chương trình chính

o. Cách truyền tham số cho hàm

4.19. Bài tập thực hành:

I. Xây dựng thủ tục cho các bài tập sau:

Bài 1: Tìm lỗi sai:

```
Var i, a: integer;
procedure thutuc1(int a);
void main()
{
        a:=4;
        for i:=0 to 5 do
                Ham1(a);
}
Procedure thutuc1(int a)
{
        for i:=0 to 5 do
                writeln('a= ',a);
}
```

Bài 2: Viết hàm kiểm tra số nguyên n có phải là **số nguyên tố** hay không?

Bài 3: Viết hàm kiểm tra số nguyên n có phải là **số chính phương** hay không? ($25=5^2$, $16=4^2$, $9=3^2$; các số 25, 16, 9 là những số chính phương)

Bài 4: Viết hàm tính n! (n là số nguyên nhập vào từ bàn phím)

Bài 5: Viết hàm tìm **UCLN** và **BCNN** của hai số nhập vào từ bàn phím

4.20. Hướng dẫn

Bài 1: Sinh viên tự làm

Bài 2:

❖Thuật toán:

p. **Viết thủ tục Procedure ktra_ngto(int x):** kiểm tra x có phải là số nguyên tố hay không
   a. Khai báo biến i, ktra
   b. Cho biến đếm i chạy từ 2 đến x
      i. Nếu x mod i = 0 thì thoát
   c. Kiểm tra nếu i=x thì x là số nguyên tố, ngược lại thì x không phải là số nguyên tố

q. **Chương trình chính**:
   a. Khai báo biến n, kq
   b. Nhập số n
   c. Gọi thủ tục với số giá trị n vừa nhập vào: **ktra_ngto(n)**

❖Chương trình:

Bài 5:

❖Thuật toán:

r. **Viết thủ tục Procedure USCLN(int a, int b):** tìm ước số chung lớn nhất của hai số a và b
   a. Lấy trị tuyệt đối hai số a và b
   b. Chừng nào (a!=0 và b!=0) làm
      i. Nếu a>b thì a=a-b
      ii. Ngược lại, b=b-a
   c. Nếu a = = 0 thì b là USCLN
   d. Ngược lại, a là USCLN

s. **Viết thủ tục Procedure BSCNN(int a, int b):** tìm bội số chung nhỏ nhất của hai số a và b
   a. BSCNN= (a*b)/USCLN(a,b)

t. **Chương trình**:
   a. Khai báo biến a, b, US, BS
   b. Nhập hai số a,b
   c. Gọi thủ tục:
      i. USCLN(a, b)
      ii. BSCNN(a, b)
   d. Xuất BS, US ra màn hình

❖Chương trình:

II. Xây dựng hàm cho các bài tập sau:

Bài 1: Viết một **hàm tính tổng các chữ số** của một số nguyên. Viết chương trình nhập vào một số nguyên, dùng hàm trên **kiểm tra** xem số **đó có chia hết cho 3** không (Một số chia hết cho 3 khi tổng các chữ số của nó chia hết cho 3).

Bài 2: Viết hàm tìm tất cả các **ước số** của số nguyên n (n nhập vào từ bàn phím). In ra các ước số đó và đếm có bao nhiêu ước số

Bài 3: Viết chương trình tính tổng sau:

- $S=1+x+x^2+x^3+\ldots+x^n$
- $S= 1 - x +x^2- x^3+ ... (-1)^n x^n$
- $S=1+ 1/2!+\ldots+1/n!$

Bài 1:

❖Thuật toán:

u. **Viết hàm (function) tong_cacchuso(int n):** Tính tổng các chữ số của số nguyên n
   a. Khai báo biến S, t
   b. Gán S=0
   c. Chừng nào n>0 làm

i. t=t%10
ii. n=n/10
iii. S=S+t

d. Hàm tong_cacchuso(int n) trả về giá trị S

v. **Chương trình chính**:

a. Khai báo biến n, tong
b. Nhập số nguyên n từ bàn phím
c. Gọi hàm:

i. tong = **tong_cacchuso(n)**
ii. Nếu tong%3= =0 thì n chia hết cho 3

❖Chương trình:

Bài 2:

❖Thuật toán:

w. **Viết hàm (function) tim_uoc_so(int n):** Tìm ước số của số nguyên n

a. Khai báo biến i, count
b. Gán count=0;
c. Cho i chạy từ 1 đến n

i. Nếu n%i= = 0 thì

1. in i ra màn hình
2. count=count+1

d. Xuất count ra màn hình

x. **Chương trình chính**:

a. Khai báo biến n
b. Nhập số nguyên n từ bàn phím
c. Gọi hàm: **tim_uoc_so(n)**

❖Chương trình:

Bài 3a:

❖Thuật toán:

y. **Viết hàm (function) tong_luythua(int x, int n):** Tìm ước số của số nguyên n

a. Khai báo biến i, S
b. Gán S=1
c. Cho i chạy từ 1 đến n

i. S=S+pow(x,i)

d. Hàm tong_luythua(int x, int n) trả về giá trị S

z. **Chương trình chính**:

a. Khai báo biến x, n, tongS
b. Nhập số nguyên x, n từ bàn phím
c. Gọi hàm: tongS = **tong_luythua(x, n)**
d. Xuất tongS ra màn hình

❖Chương trình: Sinh viên tự viết chương trình dựa trên thuật toán đưa ra

Bài 3b: Sinh viên tự làm (tham khảo câu a)

Bài 3c:

❖Thuật toán:

aa. **Viết hàm (function) giai_thua(int n):** Tính n giai thừa (n là số nguyên dương)
   a. Khai báo biến i, gth
   b. Gán gth=1
   c. Cho i chạy từ 1 đến n
      i. gth=gth*i
   d. Hàm giai_thua(int n) trả về giá trị gth

bb. **Chương trình chính**:
   a. Khai báo biến i, n, tong_gt
   b. Gán tong_gt=0
   c. Nhập số nguyên n từ bàn phím
   d. Cho i chạy từ 1 đến n
      i. Gọi hàm: tong_gt = tong_gt+1/**giai_thua(i)**
   e. Xuất tong_gt ra màn hình

❖Chương trình: Sinh viên tự viết chương trình dựa trên thuật toán đưa ra

CHƯƠNG 5:

Tên chương : Dữ liệu kiểu tập hợp, mảng và bản ghi
Mã chương : MH15-05
Mục tiêu :

- Trình bày được khái niệm tập hợp, mảng và bản ghi;
- Thực hiện cách khai báo, gán giá trị cho tập hợp, mảng, bản ghi;
- Thực hiện các phép toán trên tập hợp, mảng và bản ghi.
- Thực hiện các thao tác an toàn với máy tính.

5.1. Kiểu tập hợp, các phép toán trên tập hợp

5.2. Định nghiã

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

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);
```

Chú y:

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

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

Kết quả khi biên 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].
- 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}
```

5.3. Cac phep toan trên tập hợp

5.4. Phép gán

Ta có thể gán giá trị các tập đã được mô tả vào các biến tập cùng kiểu. Riêng tập hợp rỗng có thể gán cho mọi biến kiểu tập hợp khác nhau.

Với ví dụ trên, ta có thể gán :

```
Chu   := [X,Y,Z] ;
So    := [2,3,4] ;
Date  := [] ;
```

Nếu ta viết Chu := [1,2]; thì không hợp lệ vì Chu là tập hợp các chữ.

5.5. Phép hợp

Hợp của 2 tập hợp A và B là một tập hợp chứa tất cả các phần tử của tập A hoặc B hoặc cả A và B.

Ký hiệu của phép hợp là dấu cộng (+). Phép hợp có tính giao hoán:

A+B = B+A

Ta có thể mô tả phép hợp qua hình ảnh sau :

Minh họa phép tập hợp (phần hợp là hình A+B)

Ví dụ 8.32

```
A := [0,1,3,5,7,9] ;
B :=  [0,2,4,6,8,9] ;
C := A + B ;
```

{tập hợp C sẽ có các phần tử là [0,1,2,3,4,5,6,7,8,9] }

5.6. Phép giao

Giao của 2 tập hợp A và B là một tập chứa các phần tử của cả A và cả B.

Ký hiệu A * B. Phép giao cũng có tính giao hoán, nghĩa là A * B = B * A

Minh họa như sau :

Minh họa phép giao (phần giao là phần màu đen)

Với ví dụ trong phép hợp, nếu:

D := A * B ; {tập D chứa phần tử [0,9] }

Nếu A và B không có phần tử nào giống nhau thì phép hợp sẽ cho tập rỗng.

5.7. Phép hiệu

Hiệu của 2 tập hợp A và B, ký hiệu là A - B, là một tập hợp chứa các phần tử chỉ thuộc A mà không thuộc B. Lưu ý : A - B thì khác B - A.

Ví dụ 8.33: A := [3 .. 7] ;
B := [1.. 6, 10, 15] ;
thì A - B là tập hợp [7] còn B - A là tập hợp [1,2, 10,15]

5.8. Phép thuộc IN

Phép thuộc IN cho phép thử xem một giá trị nào đó thuộc về một tập hay không? Phép thuộc IN cho kết quả có kiểu Boolean. Nếu đúng nó sẽ cho kết quả là TRUE, ngược lại là FALSE.

Ví dụ 8.34: Chu là biến kiểu Char, còn A là biến kiểu SET OF Char và
Chu := 'X' ;
A := ['X', 'x','Y', 'y', 'Z', 'z'] ;
thì phép toán Chu IN A sẽ cho kết quả là TRUE

5.9. Các phép so sánh =, <>, <= và >=

Muốn so sánh 2 tập hợp với nhau thì chúng phải có cùng kiểu cơ bản. Kết quả của các phép so sánh là giá trị kiểu Boolean, tức là TRUE (Đúng) hoặc FALSE (Sai).

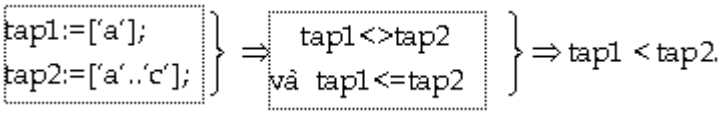
Hai tập hợp A và B gọi là bằng nhau (A = B) chỉ khi chúng có các phần tử giống với nhau từng đôi một (không kể thứ tự sắp xếp các phần tử trong 2 tập). Ngược lại của phép so sánh bằng nhau (=) là phép so sánh khác nhau (<>). Nghĩa là, nếu A = B là TRUE thì A <> B sẽ là FALSE và ngược lại.

Phép so sánh nhỏ hơn hoặc bằng (<=) của A <= B sẽ cho kết quả là TRUE nếu mọi phần tử có trong A đều có trong B. Định nghĩa này cũng tương tự như lớn hơn hoặc *bằng (>=). Với A >= B thì mọi phần tử của B đều có trong A, kết quả này TRUE,* ngược lại là FALSE.

Chú ý: Trong Pascal không có phép so sánh nhỏ hơn (<) và lớn hơn (>). Để kiểm tra xem tập A có thực sự nằm trong tập B hay không (A nhỏ hơn B), ta phải sử dụng thêm các phép logic như sau:

IF (A <> B) AND (A <= B) THEN WRITELN ('A < B')

Ví dụ về so sánh tập hợp:



5.10. Mảng một chiều

5.11. Khái niệm:

Một mảng dữ liệu là một tập hợp số hữu hạn phần tử có giống như các biến, có cùng kiểu, gọi là kiểu cơ bản. Mảng được được tổ chức theo một trật tự xác định. Số phần tử của mảng được khai báo ngay từ khi định nghĩa ra mảng.

5.12. Mảng một chiều:

Mảng một chiều có thể được hiểu như một danh sách các phần tử (theo cột), có cùng kiểu. Mỗi phần tử của mảng được xác định được truy nhập trực tiếp thông qua tên mảng cùng với chỉ dẫn truy nhập được để giữa hai ngoặc vuông [ ].

5.13. Khai báo mảng một chiều:

5.14. Khai báo gián tiếp:

TYPE

<Kiểu mảng> = ARRAY [Kiểu chỉ số] OF <Kiểu phần tử > ;

VAR

<Danh sách biến> : Kiểu mảng ;

5.15. Khai báo trực tiếp :

VAR

< Danh sách biến > : ARRAY [Kiểu chỉ số] OF < Kiểu phần tử > ;

Ví dụ:

```
TYPE
        KM1 = ARRAY [1.. 100] OF  INTEGER ;
        KM2 = ARRAY [1 .. 20 ] OF CHAR ;
        DAY = (Sun, Mon, Tue, Wed, Thu, Fri, Sat) ;
VAR
        TUOI  : KM1 ;
        TEN : KM2 ;
        NGAY : ARRAY [DAY] OF BOOLEAN ;
```

Ý nghĩa:

- KM1 là kiểu mảng gồm 100 phần tử được đánh số từ 1 đến 100 thông qua kiểu chỉ dẫn là một miền con các số nguyên từ 1 .. 100. TUOI là biến có kiểu là KM1.

- KM2 là kiểu mảng gồm 20 phần tử đánh số từ 1 .. 20 có kiểu là các ký tự. Biến TEN có kiểu là KM2.
- NGAY là một biến mảng gồm 7 phần tử kiểu Boolean được đánh dấu qua kiểu chỉ dẫn là tên của 7 ngày trong tuần.

Chú ý:

Khi khai báo mảng, kiểu chỉ dẫn chỉ có thể là:

- Kiểu miền con của các loại dữ liệu vô hướng đếm được như ký tự, số nguyên
- Kiểu liệt kê do người viết định nghĩa (như NGAY trong tuần)
- Kiểu Boolean

Kiểu chỉ dẫn không thể là kiểu không đếm được như REAL

Viết như sau là SAI : X1 : ARRAY [Real] OF Integer ;

Ta cũng không thể khai báo như: X2 : ARRAY [Integer] OF Integer ;

Mặc dầu Integer là kiểu vô hướng đếm được do giới hạn của vùng nhớ dành cho dữ liệu, số lượng phần tử của 1 mảng cũng bị hạn chế tùy theo kích thước của kiểu dữ liệu của các phần tử, ta nên dùng kiểu miền con để khai báo số phần tử của mảng.

5.16. Truy nhập và truy xuất các phần tử của mảng một chiều:

Mỗi phần tử của mảng được truy xuất thông qua tên biến mảng cùng với chỉ số của mảng trong cặp dấu ngoặc vuông [ ]. Xét ví dụ dưới đây:

```
Type  MANG = Array[1..10] of Integer;
Var   A: MANG;
```

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]

Ví dụ:

Viết chương trình nhập vào một mảng và in ra mảng đó sau khi xắp xếp các phần tử của mảng tăng dần.

```
USE   Crt;
Type  MANG = Array[1..50] of Integer;
Var   A: MANG;
      i, j, n, tam: Integer;
Begin
      Write("Bạn nhập bao nhiêu phần tử: "); Readln(n);
      {Nhập n phần tử}
      For i :=1 to n do
            Begin
                  Write("Phần tử" , i, " là: ");
                  Readln(A[i]);
            End
```

```
{Sắp xếp tăng dần }
for i := 1 to n-1 do
for j := i + 1 to n-1 do
                    If A[i] > A[j] then
                        Begin
                            tam := A[i];
                            A[i] := A[j];
                            A[j] := tam;
                    End
{In các phần tử của mảng}
for i := 1 to n do
        Write(A[i]:10);
Readln;
End.
```

Chú y: Hai mảng A và B có cùng số phần tử và cùng kiểu phần tử, ta có thể thay toàn bộ phần tử A bởi các phần tử tương ứng của B bằng một phép gán A := B.

5.17. Mảng nhiều chiều

5.18. Khái niệm

Trong phần này ta chỉ xét mảng 2 chiều. Mảng hai chiều cũng tương tự như khái niệm về ma trận.

5.19. Khai báo

ARRAY[<Tập chỉ số 1> , <Tập chỉ số2>] OF <Kiểu phần tử>;

Ví dụ:

```
Type  MANG = Array[1..20, 1..20] of Integer;
Var   A: MANG;
```

Hoặc khai báo:

```
Var   A: Array[1..20, 1..20] of Integer;
```

Mảng hai chiều có thể khai báo như là mảng một chiều của mảng một chiều, ta có thể khai báo như sau:

```
Type  KieuPhantu = Array[1..20] of Integer;
Var   A: Array[1..20] of KieuPhantu;
```

5.20. Truy nhập và truy xuất của mảng

Mảng hai chiều tổ chức như một ma trận, các phần tử của ma trận cũng tương tự như các phần tử của mảng hai chiều. Ta truy xuất các phần tử của mảng hai chiều thông qua tên biến, theo sau là cặp chỉ số cách nhau bởi dấu , (phẩy) hoặc hai cặp dấu [ ][ ].

Ví dụ:

A[3, 2] hoặc A[2][3].

Ta có thể hình dung mảng A: Array[1..4, 1..5] như sau:

A[1,1]	A[1,2]	A[1,3]	A[1,4]	A[1,5]
A[2,1]	A[2,2]	A[2,3]	A[2,4]	A[2,5]
A[3,1]	A[3,2]	A[3,3]	A[3,4]	A[3,5]
A[4,1]	A[4,2]	A[4,3]	A[4,4]	A[4,5]

Ví dụ: nhập vào một ma trận số nguyên rồi in ra ma trận đó theo dạng toán học.

```
Type MANG=array[1..20,1..20] of integer;
Var    A:MANG;
       i,j,m,n:integer;
Begin
Write('Ma tran co bao nhieu dong :')  ;readln(n);
Write('Ma tran co bao nhieu cot :')     ;readln(m);
{Nhập vào mảng hai chiều}
for i:=1 to n do
       for j:=1 to m do
       Begin
       Write('phan tu A[',i,',',j,'] la :');
       Readln(A[i,j]);
       End;
{In các phần tử ra như một ma trận}
for i:=1 to n do
       Begin
       for j:=1 to m do
       Write(A[i,j]);
       Writeln;
       End;
Readln;
End.
```

5.21. Kiểu bản ghi

5.22. Khai niệm

Hàng ngày chúng ta rất quen thuộc với một danh sách sinh viên như dưới đây:

Stt	Họ và tên	Mã số	Toán	Lý	Đtb
1	Nguyen Van An	1973208	8.0	7.0	7.5
2	Tran Thi Nga	1974564	5.0	8.0	6.5
...	...	...	...	...	...

Mỗi dòng liệt kê các dữ liệu về một người, mỗi cột là một dữ liệu thành phần cung cấp thông tin về một thuộc tính cụ thể của những người đó.

Trong ngôn ngữ Pascal, mỗi dòng được gọi là một RECORD (dịch là *bản ghi* hay *thẻ ghi*), mỗi cột là một FIELD (dịch là *trường* hay *thành phần*, hay *thuộc tính* cho sát với thực tế).

Nói tổng quát, mỗi bản ghi là một tập gồm nhiều *trường (field)*, các *trường* có thể có kiểu dữ liệu khác nhau.

Kiểu bản ghi được mô tả bằng cách dùng từ khóa RECORD kèm theo một danh sách khai báo các tên trường và kiểu dữ liệu tương ứng, kết thúc bằng từ khóa END; , tức là:

TYPE

Tênkiểu = **RECORD**

Têntrường1 : Kiểudliệu1;

Têntrường2 : Kiểudliệu2;

...

Têntrườngk : Kiểudliệuk;

End;

- Ví dụ 1: Ta định nghĩa một kiểu KSVIEN như sau:

TYPE

KSVIEN = **RECORD**

Hoten:String[20];

Maso : String[8];

Toan, Ly, DTB: Real;

End;

Theo mô tả trên, ta có một kiểu dữ liệu mới đặt tên là KSVIEN có cấu trúc bản ghi gồm 5 trường (thuộc tính) là:

Tên trường	Kiểu dữ liệu	Ý nghĩa
Hoten	String[20]	Họ và tên của sinh viên
Maso	String[8]	Mã số sinh viên
Toan	Real	Điểm toán của sinh viên
Lý	Real	Điểm lý của sinh viên
DTB	Real	Điểm trung bình của sinh viên

Ví dụ 2:

Ta mô tả thời gian là kiểu KDATE có ba trường ngày , tháng, năm như sau:

TYPE

KDATE = **RECORD**

Ngay : 1..31;

Thang : 1..12;

Nam : Integer;

End;

Ví dụ 3:

Để quản lý các sách trong một thư viện, ta xây dựng một kiểu bản ghi KSACH như sau:

TYPE

KSACH = RECORD

Ma_so_sach: String[6];

Ten_doc_gia: String[20];

Nam_xban :Integer;

Gia_tien: Real;

Ngay_muon : KDATE;

End;

Kiểu KSACH là một bản ghi có 5 trường mô tả 5 thuộc tính của sách là: mã số sách, tên độc giả, năm xuất bản, gía tiền và ngày mượn.

Ví dụ này cho thấy các bản ghi có thể được mô tả lồng nhau: kiểu dữ liệu của một trường của bản ghi này lại có thể là một kiểu bản ghi khác đã được định nghĩa trước đó. Trong bản ghi KSACH, Ngay_muon là một trường có kiểu dữ liệu là một bản ghi kiểu KDATE.

Mỗi đối tượng cần quản lý có thể có nhiều trường, song tùy yêu cầu quản lý mà ta chỉ lựa chọn và khai báo những trường thật sự cần thiết. Khai báo thừa thì hao phí bộ nhớ, nhưng nếu thiếu thì công tác quản lý sẽ khó khăn do thiếu thông tin. Vì vậy, nên khai báo các trường với số lượng ít nhất nhưng đủ dùng.

5.23. Sử dụng bản ghi

Kiểu bản ghi sau khi đã được định nghĩa có thể dùng khai báo cho các biến. Ví dụ :

Var

X, Y, Z : **KSVIEN;**

Trong đó KSVIEN là bản ghi đã mô tả ở phần trên.

Theo khai báo này, X ,Y và Z là ba biến kiểu bản ghi KSVIEN, mỗi biến đều có 5 trường Hoten, Maso, Toan, Ly và DTB.

Để thâm nhập vào một trường của bản ghi ta viết tên biến kiểu bản ghi, sau đó là dấu chấm '**.**' và tên trường, tức là :

Tênbiến .Têntrường

Các lệnh dưới đây gán gía trị cho từng trường của biến X :

X.Hoten :='Nguyen Van An';

X.Maso :='1973208';

X.Toan :=8.0;

X.Ly :=7.0;

X.DTB :=(X.Toan+X.Ly)/2;

Để nhập dữ liệu cho trường Hoten của biến Y, ta viết:

Write('Nhap ho ten sinh vien Y : ');

Readln(Y.Hoten);

Sở dĩ phải viết tên biến bản ghi ở trước tên trường là để xác định trường đó là của biến bản ghi nào. Mỗi biến X, Y, Z đều có trường Hoten, nên nếu chỉ viết Hoten thôi thì không biết đó là Hoten của ai: X, Y hay Z ?. Còn viết X**.**Hoten là chỉ rõ đây là Hoten của biến X.

Như vậy, mỗi trường của biến bản ghi có thể thâm nhập và sử dụng như một biến bình thường. X.Hoten là biến kiểu String[20], X.Maso là biến kiểu String[8], ..., X.DTB là biến kiểu Real.

Đối với các bản ghi lồng nhau, cách truy xuất đến từng trường cũng tương tự.

Ví dụ: Cho khai báo biến S kiểu KSACH:

Var

S : KSACH ;

Để truy nhập đến các trường Ngay, Thang, Nam của Ngay_muon ta viết :

S**.**Ngay_muon**.**Têntrường

Chẳng hạn gán :

S.Ngay_muon.Ngay := 2;

S.Ngay_muon.Thang := 9;

S.Ngay_muon.Nam := 1999;

Hai biến bản ghi cùng kiểu có thể gán cho nhau. Lệnh :

Y:=X;

gán gía trị của từng trường của biến X cho trường tương ứng của biến Y. Vậy lệnh trên tương đương với khối 5 lệnh sau :

begin

Y.Hoten :=X.Hoten;

Y.Maso :=X.Maso;

Y.Toan :=X.Toan;

Y.Ly :=X.Ly;

Y.DTB :=X.DTB;

end;

Các bản ghi có thể so sánh bằng nhau hoặc khác nhau:

Ví dụ:

If X=Y then writeln(‘ X và Y là một người ‘);

If X<>Y then writeln(‘ X khác Y ‘);

Tuy nhiên không có phép so sánh <, <=, >, >= cho các bản ghi.

Hai bản ghi có thể hoán đổi gía trị cho nhau theo nghĩa hoán đổi từng cặp gía trị của các trường tương ứng.

Giống như các biến đơn giản, để hoán đổi hai bản ghi X và Y ta dùng ba lệnh:

Z:=X; X:=Y; Y:=Z;

trong đó Z là biến trung gian cùng kiểu bản ghi với X và Y.

Ví dụ: Nếu biến X và Y có các trường tương ứng là :

X.Hoten =‘Nguyen Van An’ X.Maso =‘1973208’	Y.Hoten =‘Tran Thi Nga’ Y.Maso =‘1974564’

X.Toan =8.0 X.Ly =7.0 X.DTB =7.5	Y.Toan =5.0 Y.Ly =8.0 Y.DTB =6.5

thì sau khi hoán đổi, ta được :

X.Hoten =‘Tran Thi Nga’ X.Maso =‘1974564’ X.Toan =5.0 X.Ly =8.0 X.DTB =6.5	Y.Hoten =‘Nguyen Van An’ Y.Maso =‘1973208’ Y.Toan =8.0 Y.Ly =7.0 Y.DTB =7.5

5.24. Câu lệnh WITH

Khi làm việc với nhiều trường của một biến bản ghi thì cách thâm nhập ở trên tỏ ra rườm rà vì phải viết nhiều lần tên biến trước tên trường.

Để đơn giản cách viết, Pascal đưa ra câu lệnh :

WITH Tênbiến DO Lệnh;

Tên biến thuộc kiểu bản ghi.

Nếu LệnhP có truy xuất đến các trường của Tên biến thì không cần phải viết Tên biến và dấu chấm trước các tên trường.

Ví dụ, thay vì viết:

```
X.Hoten:= ‘Nguyen Van An’;
```

ta có thể viết :

```
WITH X DO Hoten:= ‘Nguyen Van An’;
```

Để in các trường của biến X lên màn hình, ta viết:

```
 WITH X DO
Begin
```

```
    Writeln(‘ Họ và tên :’ , Hoten);
    Writeln(‘ Mã số sinh viên :’ , Maso);
    Writeln(‘ Điểm Toán :’ , Toan: 4:1);
    Writeln(‘ Điểm Lý :’ , Ly: 4:1);
    Writeln(‘ Điểm trung bình : ‘ , DTB :4:1);
End;
```

Tất cả các tên trường nằm trong khối begin và end được hiểu là các trường của biến X (nếu không ghi rõ tên biến nào khác).

Các lệnh sau gán các gía trị cho các trường của biến S kiểu KSACH là một bản ghi lồng nhau:

```
WITH S DO
 begin
  Ma_so_sach:=‘TH-435’;
  Ten_doc_gia:=‘Nguyen van Mai’;
  Nam_xban :=1999;
  Gia_tien:= 15000;
  WITH Ngay_muon DO
   begin
    Ngay:=2;
    Thang:=9;
    Nam:=1999;
   end;
  end;
```

5.25. Bản ghi co´câú truc thay đổi

Các kiểu Record trình bày trên là kiểu Record cố định vì số thanh phần cũng như cấu trúc cua 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 ví 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 nay 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.
```

Giải thích :

- 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 đôi 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).

Lưu ý :

- Phần thay đổi là một trường gọi là trường đánh dấu (Tag Field) và được đặt 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.

5.26. Bài tập:

Nội dung

cc. Cách thức khai báo và truy xuất các phần tử trong **mảng một chiều**

dd. Cách thức khai báo và truy xuất các phần tử trong **mảng nhiều chiều** (cụ thể là mảng hai chiều)

5.27. Bài tập thực hành:

5.28. Mảng một chiều

Bài 1: Viết chương trình nhập vào mảng một chiều gồm n phần tử. Sau đó in ra các phần tử đã nhập lên màn hình

Bài 2: Viết chương trình nhập vào một mảng A, hãy xuất ra màn hình:

- Phần tử lớn nhất của mảng.
- Phần tử nhỏ nhất của mảng.
- Tính tổng của các phần tử trong mảng .

Bài 3: Viết chương trình nhập vào mảng một chiều A gồm n phần tử. Xuất ra màn hình các yêu cầu sau:

ee. **Sắp xếp** các phần tử trong mảng theo **thứ tự tăng dần**,

ff. **Đảo mảng** đã sắp xếp trên

gg. **Tính tổng và tích** các số nguyên dương

Bài 4: Viết thủ tục nhập vào mảng A gồm n phần tử kiểu nguyên, rồi viết hàm tìm kiếm x có trong mảng A hay không?

Bài 5: Viết chương trình nhập vào một mảng số tự nhiên. Hãy xuất ra màn hình:

- Dòng 1: gồm các số lẻ, tổng cộng có bao nhiêu số lẻ.
- Dòng 2: gồm các số chẵn, tổng cộng có bao nhiêu số chẵn.
- Dòng 3: gồm các số nguyên tố.
- Dòng 4: gồm các số không phải là số nguyên tố.

Bài 6: Viết chương trình nhập vào một dãy các số theo thứ tự tăng, nếu nhập sai quy cách thì yêu cầu nhập lại. In dãy số sau khi đã nhập xong. Nhập thêm một số mới và chèn số đó vào dãy đã có sao cho dãy vẫn đảm bảo thứ tự tăng. In lại dãy số để kiểm tra.

Bài 7: Viết chương trình tính tổng bình phương của các số âm trong một mảng các số nguyên.

5.29. Mảng hai chiều

Bài 1: Viết chương trình nhập vào ma trận cấp n x m. Sau đó xuất ra màn hình ma trận đã nhập

Bài 2: Viết chương trình nhập vào ma trận cấp n x m bằng phương pháp Random. Sau đó xuất ra màn hình ma trận đã nhập

Bài 3: Viết chương trình nhập vào ma trận cấp n x m. Sau đó tìm xem cột nào có nhiều số dương nhất, rồi xuất kết quả ra màn hình

Bài 4: Viết chương trình nhập vào ma trận cấp n x m. Thực hiện chuyển vị ma trận, in ra màn hình ma trận đã chuyển vị.

Bài 5: Viết chương trình nhập vào hai ma trận A và B có cấp m, n. In hai ma trận lên màn hình. Tổng hai ma trận A và B là ma trận C được tính bởi công thức:

cij= aij +bij (i=0,1,2,...m-1; j=0,1,2...n-1)

Tính ma trận tổng C và in kết quả lên màn hình.

5.30. Bản ghi

Bài 1: Viết chương trình thực hiện phép cộng 2 số phức.

Bài 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.

Bài 3: 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.

d/ Tìm và in ra màn hình tất cả các cuốn sách có cùng tác giả được nhập vào từ bàn phím.

e/ Lọc ra các cuốn sách được xuất bản trong cùng một năm nào đó.

CHƯƠNG 6:

Tên chương :Dữ liệu kiểu chuỗi
Mã chương: MH15-06
Mục tiêu:

- Trình bày được khái niệm dữ liệu kiểu chuỗi kí tự ;
- Biết sử dụng dữ liệu kiểu chuỗi trong chương trình ;
- Áp dụng được các phép toán trên chuỗi ;
- Vận dụng được các hàm xử lý chuỗi để xử lý.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

Chuỗi (String) là kiểu dữ liệu có cấu trúc dùng để xử lý các chuỗi 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 chuỗi có thể thay đổ còn số phần tử của kiểu mảng luôn cố định.

6.1. Khai báo và các phép toán

6.2. Khai baó kiểu chuôĩ

VAR Tên_Biến : String[n];

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

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 chuỗi, 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[4]) sẽ in ra ký tự ‘D’.
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 chuỗi đó có độ dài bao nhiêu.
Chẳng hạn biến HoTen bên trên được gán giá trị:
HoTen := ‘Nguyen Van An’;

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

0	1	2	3	4	5	6	7	8	9	10	11	12	13				30
Chr(13)	N	g	u	y	e	n		V	a	n		A	n	….	*	….	*

*Chú ý: * chỉ ký tự không xác định*

6.3. Cac phep toan trên chuôĩ

6.4. Phép gán

Biến := Biểu_thức;

Đại lương bên phải của lệnh phải được đặt giữa hai dấu nháy đơn nếu đó là chuỗi ở dạng hằng. Ta có thể sử dụng dấu cộng (+) để ghép các chuỗi khi gán.

Ví dụ: HoTen := ‘Nguyen Van‘ + ‘ An‘;

6.5. Phép nối chuỗi

Ký hiệu bằng dấu +.

Ví dụ: ‘Turbo’ + ‘ Pascal‘ = ‘Turbo Pascal‘

6.6. Các phép toán so sánh

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

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.

6.7. Nhập va`in chuôĩ

Muốn in một chuỗi ký tự ta dùng lệnh Write(St) hoặc Writeln(St). Lệnh Readln(St) sẽ đọc các ký tự cho chuỗi 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 chuỗi rỗng.

6.8. Các thủ tục va`hàm làm việc với chuỗi

6.9. Các thủ tục

6.10. Insert(Obj, St, Pos):

Chèn vào chuỗi ký tự **St** một chuỗi ký tự **Obj** kể từ vị trí **Pos**.

***Ví dụ*:**

St := ‘Thh’;

Insert(‘an’, St, 3);

Writeln('Xau ky tu sau khi chen them la: ', St); {Sẽ là 'Thanh'}

6.11. Delete(St, Pos, Num):

Xóa bỏ trong chuỗi ký tự **St** bắt đầu từ vị trí thứ **Pos** một số ký tự là **Num.**

***Ví dụ*:**

```
St := 'Truong Hoc';
Delete(St, 4, 6);
Writeln('Xau ky tu sau khi xoa bot la: ', St);  {Sẽ là 'Truc'}
```

6.12. Thủ tục Str(Value, St):

Sẽ biến đổi giá trị bằng số nguyên hoặc số thực **Value** thành một dãy ký tự biểu diễn số đó. Cách biểu diễn của **St** sẽ được qui định do qui cách của **Value**.

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' }
```

6.13. Thủ tục Val(St, Var1, Code):

Sẽ biến đổi chuỗi ký tự **St** (*biểu diễn một số nguyên hay số thực*) thành một số nguyên hay số thực chứa trong **Var1**. *Với*: **Code** là số nguyên để phát hiện lỗi:

- **Code = 0** tức phép biến đổi là đúng.
- Nếu **St** không biểu diễn đúng số nguyên hay số thực thì **Code** nhận giá trị bằng vị trí của ký tự sai trong chuỗi **St** (**Code <> 0**).

***Ví dụ*:**

X là biến thực, St = '123.56', Result là biến nguyên.

```
Val(St, X, Result);
Kết qủa: X = 123.56      và Result = 0
```

X là biến nguyên, St = '123', Result là biến nguyên.

```
Val(St, X, Result);
Kết qủa: X = 123         và Result = 0
```

X là biến thực, St = '123.56A', Result là biến nguyên.

```
Val(St, X, Result);  {St biểu diễn không đúng}
Kết qủa: X không xác định  và Result = 7
```

6.14. Các hàm

6.15. Hàm Lenght(St):

Kết qủa nhận được là độ dài của chuỗi ký tự St.

Ta có thể tự viết lại hàm Length như sau:

```
Function Length(St: String): Byte;
Begin
  Length := Ord(St[0]);
End;
```

6.16. Hàm Copy(St, Pos, Size):

Sao chép ở trong chuỗi lý tự **St** bắt đầu từ vị trí là **Pos** và nhận **Size** ký tự.

Ví dụ: St = 'ABCDEF' thì lệnh Copy(St, 3, 2) = 'CD' và Copy(St, 4, 10) cho ta 'DEF

✓ **Ghi chú:**

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

6.17. Concat(St1, St2,..., Stn):

Hàm này dùng để ghép tất cả các chuỗi ký tự St1, St2,..., Stn thành một chuỗi theo thứ tự các đối số cung cấp cho hàm.

✓ **Ghi chú:**

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

6.18. Pos(St1, St2):

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

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

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

6.19. BÀI TẬP CHƯƠNG 6

Câu 1. Viết thủ tục để kiểm tra và nhập một chuỗi ký tự số nguyên S, tính tổng các chữ số của chuỗi S, rồi đảo chuỗi.

Câu 2. Viết chương trình đổi ký tự chữ hoa ra chữ thường.

Câu 3. viết chương trình cắt các ký tự trống ở bên trái chuỗi ký tự.

Câu 4. Viết chương trình cắt các ký tự trống ở bên phải chuỗi ký tự.

Câu 5. Viết chương trình cắt các ký tự trống ở giữa chuỗi ký tự.

Câu 6. Viết chương trình kiểm tra một chuỗi nhập vào từ bàn phím có đối xứng hay không. Ví dụ:

- Chuỗi 'ABCDCBA' là đối xứng
- Chuỗi 'ABCDABC' là không đối xứng.

Câu 7. Viết chương trình đếm số lần xuất hiện của các ký tự nào đó trong chuỗi.

Câu 8. Viết chương trình nhập vào một chuỗi từ bàn phím, hãy tách một chuỗi con gồm n ký tự từ ví trí chỉ định.

Câu 9. Viết chương trình nhập vào một câu ca dao, tục ngữ. Hãy đếm xem câu đó có bao nhiêu từ.

TÀI LIỆU THAM KHẢO

[1]. Quách Tuấn Ngọc. ***Ngôn ngữ lập trình Pascal*****. NXB Thống kê - năm 2001.**

[2]. Hoàng Hồng. ***Lập trình Turbo Pascal 7.0*****. NXB Thống kê - năm 2007.**

[3]. Bùi Thế Tâm. ***Turbo Pascal 7.0*****. NXB Giao thông vận tải - năm 2006.**

[4]. Nguyễn Đình Tê. Borland Pascal 7.0 . NXB Lao động xã hội – năm 2007

[5]. Vũ Đức Khánh. ***Kỹ năng cơ bản trong lập trình Pascal*** **. NXB Văn hóa thông tin – năm 2010.**

[6]. Nhiều tác giả. Giáo trình ***Turbo Pascal*** **. NXB Giao thông vận tải - năm 2007.**

DANH SÁCH BAN BIÊN SOẠN GIÁO TRÌNH DẠY NGHỀ
TRÌNH ĐỘ CAO ĐẲNG

Tên giáo trình: Lập trình căn bản
Tên nghề: Quản trị mạng

1. Bà Võ Thị Ngọc Tú	Chủ nhiệm
2. Bà Trần Thị Hà Khuê	Thành viên
3. Bà Đặng Quý Linh	Thành viên

DANH SÁCH HỘI ĐỒNG NGHIỆM THU
GIÁO TRÌNH DẠY NGHỀ TRÌNH ĐỘ TRUNG CẤP, CAO ĐẲNG

1. Ông (bà).....	Chủ tịch
2. Ông (bà).....	Phó chủ tịch
3. Ông (bà).....	Thư ký
4. Ông (bà).....	Thành viên
5. Ông(bà).....	Thành viên
6. Ông(bà).....	Thành viên
7. Ông(bà).....	Thành viên
8. Ông(bà).....	Thành viên
9. Ông(bà).....	Thành viên

Phụ lục 3.2

CÁC TIÊU CHÍ VÀ TIÊU CHUẨN ĐÁNH GIÁ CHẤT LƯỢNG GIÁO TRÌNH DẠY NGHỀ TRÌNH ĐỘ TRUNG CẤP VÀ CAO ĐẲNG

Số TT	Các tiêu chí đánh giá	Mức độ đánh giá			Ghi chú
		Đạt yêu cầu đề nghị ban hành ngay	Đạt yêu cầu nhưng phải chỉnh sửa	Chưa đạt yêu cầu phải xây dựng lại	
A	**Sự tương ứng với chương trình**				
1	Giáo trình có đủ các đề mục và thể hiện nội dung theo đúng mẫu định dạng				
2*	Giáo trình có đầy đủ các nội dung theo chương trình chi tiết các môn học/mô đun trong chương trình đào tạo				
3*	Nội dung các chương/bài đảm bảo mục tiêu kiến thức, kỹ năng đã đề ra không?				
4*	Khối lượng các thông tin trong các môn học/mô đun có phù hợp với thời lượng của chương trình không?				
B	**Tính logíc**				
5*	Nội dung từng chương/bài có được trình bày một cách logic với quá trình nhân thức không? (tức là: Mức độ từ dễ đến khó, tính trình tự cho các khái niệm từ đơn giản đến phức tạp)				
6*	Các bước hình thành kỹ năng có hợp lý và vừa phải không? (tức là quan sát mẫu - bắt trước - làm được - làm độc lập - làm thuần thục hoặc theo đường xoắn ốc để hình thành các kỹ xảo)				
7*	Mối quan hệ giữa lý thuyết và thực hành có hợp lý để bảo đảm được sự nhận thức và kiến thức, sự hình thành kỹ năng không?				
8*	Hình thức học tập và các giải pháp sư phạm cho từng chủ đề có thích hợp so với mục tiêu đã đề ra không?				
C	**Mức đầy đủ/bao quát đối với mục tiêu**				
9*	Nội dung có đầy đủ để đảm bảo đào tạo có kết quả theo các mục tiêu thực				

	hiện không				
10*	Nội dung có được nhấn mạnh để rèn luyện, hình thành các kỹ năng cần thiết không? (Tức là có các quy trình rèn luyện/thực hành bao gồm cả các khía cạnh khác như: tinh thần trách nhiệm, tuân thủ kỷ luật, ý thức an toàn, ứng xử trong nhóm, tác phong công nghiệp…)				
11*	Các cấu phần tạo sự chủ động và học tích cực có đầy đủ không? (tức là đủ các mục: Giới thiệu, hướng dẫn, tự đánh giá, giải thích thuật ngữ, tài liệu tham khảo…)				
12	Có vận dụng được sự hỗ trợ của các trang thiết bị, nguồn học liệu, nguồn l;ực khác cho quá trình học tập của học viên không?				
13	Các hành ảnh minh họa, bảng biểu, bản vẽ, quy trình thực hiện…có đủ ở mức cần thiết, rõ ràng và ăn nhập với đoạn viết không?				
D	**Tính chuẩn xác**				
14	Nội dung khoa học của thông tin có chính xác không? (về bản chất vấn đề, về các số liệu, về các sự kiện và đường nét…được đề cập trên các đoạn viêt, các bảng biểu và các hình minh họa, bản vẽ..)				
15	Các thuật ngũ có đảm bảo tính phổ thông và nhất quán không?				
E	**Phong cách biên soạn**				
16	Ý tứ trình bày rõ ràng, sáng sủa, đơn giản và dễ hiểu không?				
17	Cân đối và phù hợp giữa kênh hình và kênh chữ				
18	Có vi phạm gì về văn hóa tập quán của các dân tộc Việt Nam không?				
19	Có sai phạm gì đối với Luật bản quyền không?				
20	Phong cách trình bày có thể hiện tính gợi mở, lôi kéo người học thực hiện công việc không?				
F	**Cấu trúc và các chuyên mục**				
21	Bố cục có nhất quán trong toàn bộ tài liệu không?				
22	Mối liên hệ giữa các chuyên mục có chặt chẽ và tương ứng với nhau không?				

	(đặc biệt là mục tiêu, kiểm tra đánh giá và các hướng dẫn trả lời)				
23	Mã các chuyên mục, hình vẽ, bảng biểu, bản vẽ…có nhất quán và chính xác và tạo điều kiện thuận lợi cho việc tìm kiếm và liên hệ				

Ghi chú:

*1. Các tiêu chí có đánh dấu * có ý nghĩa rất quan trọng đối với chất lượng giáo trình đã biên soạn*

2. Các mức độ đánh giá:

- Đạt yêu cầu: Không phải sửa chữa gì hoặc chỉ cần sửa chữa vài lỗi nhỏ về biên tập;

- Đạt yêu cầu nhưng phải chỉnh sửa: Phải sửa chữa một số lỗi về nội dung chuyên môn và biên tập, chỉnh lý, bổ sung; sau đó trình chủ tịch, phó chủ tịch và thư ký hội đồng xem xét, nếu thông qua được thì đạt yêu cầu đề nghị phê duyệt;

- Không đạt yêu cầu: Có nhiều lỗi về nội dung chuyên môn và biên tập, phải biên soạn lại để trình Hội đồng thẩm định lại.