TRƯỜNG ĐẠI HỌC PHẠM VĂN ĐỒNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI GIẢNG:

NGÔN NGỮ LẬP TRÌNH PASCAL

Nguyễn Ánh

Tháng 12/2013

Mục lục

LỜI GIỚI THIỆU

Ngôn ngữ lập trình Pascal do Niklaus Wirth phát triển của ngôn ngữ Algol-60 (ngôn ngữ Euler) đưa ra vào năm 1970. Pascal là ngôn ngữ đặc biệt thích hợp cho kiểu lập trình có cấu trúc và là ngôn ngữ cho phép mô tả thuật toán rất thuận tiện. Cho đến nay, Pascal vẫn được dùng để giảng dạy về lập trình trong một số trường Phổ thông và Đại học trên thế giới.

Bài giảng này, cung cấp cho Sinh viên chuyên ngành Sư phạm tin các kiến thức cơ bản về ngôn ngữ lập trình Pascal cả lý thuyết và bài tập thực hành. Trong bài giảng nầy chúng tôi có đưa thêm một số bài tập tin học 11, để sinh viên làm quen dần với các dạng bài ở phổ thông. Nội dung này được học trong phạm vi 3 tín chỉ, phục vụ cho việc dạy và học theo phương pháp đào tạo tín chỉ.

Bài giảng bao gồm 9 chương như sau:

Chương 1. Các kiến thức chung về ngôn ngữ lập trình Pascal: Giới thiệu các thành phần cơ bản của một ngôn ngữ lập trình. Cấu trúc tổng quát của một chương trình viết bằng ngôn ngữ lập trình Pascal. Cách biên dịch và thực thi một chương trình viết bằng ngôn ngữ lập trình Pascal.

Chương 2. Các kiểu dữ liệu đơn giản: Chương nầy giới thiệu tổng quan về các kiểu dữ liệu của Pascal và đi sâu mô tả các kiểu dữ liệu đơn giản chuẩn cùng các kiểu dữ liệu do người dùng định nghĩa. Đối với mỗi kiểu dữ liệu giới thiệu cách khai báo, cách sử dụng, miền giá trị cùng các phép toán tác động lên kiểu dữ liệu đó.

Chương 3. Biểu thức và câu lệnh đơn giản: Chương nầy đưa ra các kiến thức về các thành phần tạo thành một biểu thức trong Pascal, cách tác động của các toán tử lên các toán hạng trong một biểu thức. Đồng thời giới thiệu các loại câu lệnh đơn giản, cách khai báo và sử dụng các câu lệnh này trong chương trình.

Chương 4. Câu lệnh có cấu trúc: Nội dung chương nầy đưa ra các câu lệnh có cấu trúc. Trong mỗi lệnh đưa ra cú pháp của từng lệnh. Tiếp đến phân tích cách sử dụng các câu lệnh như thế nào cho phù hợp.

Chương 5. Kiểu dữ liệu tập hợp, kiểu mảng, kiểu xâu: Nội dung chương nầy đưa ra các kiểu dữ liệu có cấu trúc. Trong mỗi kiểu dữ liệu đưa cú pháp của từng kiểu, cách định nghĩa và khai báo chúng và mô tả các thao tác trên từng kiểu dữ liệu đó.

Chương 6. Chương trình con: Chương này đưa ra cách định nghĩa một chương trình con dạng thủ tục, chương trình con dạng hàm. Đồng thời hướng dẫn cách sử dụng các chương trình con này trong một chương trình sao cho phù hợp.

Chương 7. Kiểu bản ghi: Nội dung chương này giới thiệu kiểu dữ kiểu bản ghi. Cách định nghĩa và khai báo kiểu bản ghi và các thao tác trên kiểu bản ghi.

Chương 8. Kiểu tập tin: Chương này giới thiệu kiểu dữ liệu tập tin cùng các thao tác trên tập tin định kiểu và tập tin văn bản.

Chương 9. Kiểu con trỏ: Chương này giới thiệu kiểu con trỏ là một kiểu dữ liệu động (biến động). Cách định nghĩa và khai báo kiểu con trỏ cùng các thao tác trên nó.

Để hoàn thành bài giảng này, chúng tôi xin chân thành cảm ơn Thầy Nguyễn Thanh Tiên và Thầy Nguyễn Hải Lộc đã đóng góp ý kiến và sửa chữa cho hoàn chỉnh bài giảng. Tuy nhiên, do hạn chế về thời gian, cũng như trình độ nên bài giảng chắc chắn còn nhiều sai sót. Chúng tôi mong nhận được các ý kiến đóng góp của các bạn đọc, nhằm hoàn thiện hơn nữa nội dung bài giảng này.

Nguyễn Ánh

CHƯƠNG 1. CÁC KIẾN THỨC CHUNG VỀ NGÔN NGỮ LẬP TRÌNH PASCAL

Mở đầu: Giới thiệu các kiến thức chung về ngôn ngữ lập trình Pascal. Các thành phần cơ bản của một ngôn ngữ như: Bộ ký tự, từ, từ khóa, tên, tên chuẩn. Cấu trúc tổng quát của một chương trình viết bằng ngôn ngữ lập trình Pascal. Cách biên dịch và thực thi một chương trình viết bằng ngôn ngữ lập trình Pascal.

Mục tiêu: Học xong chương này Sinh viên

- Hiểu được các khái niệm cơ bản về ngôn ngữ lập trình Pascal.
- Biết cách sử dụng Turbo Pascal 7.0.
- Biết tạo, lưu và mở tập tin chương trình.
- Biết biên dịch chương trình và thực thi chương trình.
- Biết được các ký tự được sử dụng trong NNLT Pascal.
- Biết cấu trúc của một chương trình Pascal.

1.1. Giới thiệu

Pascal là ngôn ngữ lập trình cấp cao của tác giả NIKLAUSWITH được công bố vào đầu năm 1970. (Lấy tên Pascal là để kỷ niệm nhà toán học Pascal ở thế kỷ XVII).

Ban đầu nó chỉ là một ngôn ngữ dạy học, về sau vì những ưu điểm mà nhiều hãng đã phát triển nó thành các phần mềm riêng biệt để phục vụ cho công việc riêng của mình.

Turbo Pascal là một sản phẩm của hãng Borland (Mỹ) được phát triển từ Pascal. Đây là một ngôn ngữ lập trình bậc cao có cấu trúc, được dùng phổ biến trong nước cũng như trên thế giới đặc biệt là trong lĩnh vực dạy học lập trình.

Turbo Pascal gồm các file chính sau:

- Turbo.exe: Là file chương trình soạn thảo, dịch và liên kết với bảng chọn.
- Turbo.tpl : Là file thư viện lưu các đơn vị chương trình chuẩn để chạy với Turbo.exe.
- Graph.tpu: Là file đơn vị chương trình xử lí đồ họa.
- *.chr : Là các file chứa các font chữ trong chế độ dồ họa.
- *.bgi : Là file chứa các font chữ màn hình.

Ngoài ra còn các file khác với các chức năng riêng biệt.

1.2. Các bước lập trình giải bài toán

1.2.1. Các bước lập trình tổng quát

- Bước 1: Phân tích bài toán; Xác định dữ liệu vào, dữ liệu ra làm cơ sở cho việc hình thành giải thuật.

- Bước 2: Xây dựng thuật toán : Trên cơ sở xác định dữ liệu vào, ra; các giả thiết của bài toán, các mối liên hệ giữa chúng và những kiến thức liên quan ta cần đưa ra thuật toán tương ứng.

- Bước 3: Tổ chức dữ liệu và lập chương trình theo thuật toán đã đề ra.

1.2.2. Các bước lập trình và thực hiện (với Turbo Pascal)

Bước 1: Khởi động Turbo Pascal (Turbo.exe)

Bước 2: Soạn thảo chương trình.

Bước 3: Biên dịch (nhấn tổ hợp phím ALT - F9).

- Nếu thành công thì chuyển qua bước 4.
- Nếu không thành công quay lại bước 2.

Bước 4: Chạy thử chương trình (CTRL - F9) .

- Nếu tốt chuyển qua bước 5.
- Nếu chưa tốt thì quay về bước 2.

Bước 5: Ghi vào đĩa, Dịch thành file *.exe.

Bước 6: Kết thúc.

(Mở file mới để soạn thảo chương trình khác hoặc thoát khỏi Turbo Pascal bằng việc nhấn tổ hợp phím ALT - X để thoát).

1.3. Các khái niệm và các thành phần cơ bản

1.3.1. Bộ ký tự (dùng để soạn thảo chương trình)

Bao gồm các loại ký tự sau đây:

Các chữ cái: a..z; A..Z (Tuy nhiên khi soạn thảo chương trình, Turbo Pascal không phân biệt chữ hoa hay chữ thường).

Các ký tự số: 0..9.

Các dấu toán: + - * / ^ = > <.

Các ký tự đặc biệt:? ; . : ! [ ] { } # $ @.

Dấu gạch nối: _

Các ký tự điều khiển.

Từ và từ khóa

a. Từ:

Là một dãy liên tiếp các ký tự không chứa ký tự trắng và ký tự điều khiển.

b. Từ khóa:

Là từ dành riêng của Pascal với chức năng và cú pháp được quy định sẵn. Vì vậy khi sử dụng phải theo đúng quy định, và không được sử dụng các từ khóa vào các công việc khác.

Ví dụ: Begin, end, if, then, const, var, function...

1.3.3. Tên và tên chuẩn.

a. Tên:

Tên là một từ bao gồm tối đa 255 ký tự, chỉ được lấy trong các chữ cái, chữ số, và dấu gạch nối, nhưng không được bắt đầu bằng số. Tên dùng để đặt cho các đối tượng trong chương trình như hằng, biến, hàm, thủ tục, kiểu dữ liệu...

b. Tên chuẩn:

Là tên mà Turbo Pascal đã định nghĩa sẵn để chỉ các hàm, hằng, biến, thủ tục thư viện của nó.

Chú ý: Turbo Pascal cho phép người sử dụng có thể định nghĩa lại các tên chuẩn để dùng vào các công việc khác.

1.4. Cấu trúc tổng quát của một chương trình:

Một chương trình của Turbo Pascal gồm 3 phần.

- Tiêu đề

- Khai báo và định nghĩa

- Thân chương trình

Cụ thể:

Program	Tiêu đề.
Uses *Const* *Type* *Var* *Procedure* *Function*	Các khai báo và định nghĩa
Begin *End.*	Thân chương trình

1.4.1. Phần tiêu đề

Từ khóa để khai báo là Program tiếp đến là tên của chương trình do người dùng tự đặt. Phần này không bắt buộc phải có.

(**Chú ý:** tên của chương trình phải theo đúng quy cách tên của Turbo Pascal).

1.4.2. Phần khai báo và định nghĩa:

Uses: Dùng để khai báo các Unit (đơn vị chương trình) của Turbo Pascal. Nếu có nhiều unit thì sử dụng dấu phẩy "," để ngăn cách.

Const: Dùng để khai báo các hằng.

Cú pháp: Tên_hằng = Giá trị.

Type: Dùng để định nghĩa các kiểu dữ liệu của người dùng.

Cú pháp: Tên _kiểu = định nghĩa cụ thể cho từng kiểu.

Var: Khai báo biến.

Cú pháp: Tên biến: Kiểu_dữ liệu;

(Hoặc khai báo trực tiếp không thông qua kiểu.).

Nếu có nhiều biến cùng kiểu thì sử dụng dấu phẩy “,” để ngăn cách.

Proedure: Định nghĩa chương trình con dạng thủ tục.

Function: Định nghĩa chương trình con dạng hàm.

Chú ý: Tùy thuộc vào từng chương trình cụ thể mà trong chương trình có thể có các phần khai báo và định nghĩa phù hợp, có những chương trình có phần khai báo, định nghĩa này mà không có khai báo, định nghĩa kia hoặc ngược lại, thậm chí có những chương trình không cần đến một khai báo hay định nghĩa nào cả.

1.4.3. Phần thân chương trình:

Được bắt đầu bằng từ khóa "Begin" và kết thúc bởi từ khóa "End.". Giữa cặp từ khóa này là các câu lệnh của chương trình. Nếu có nhiều câu lệnh thì sử dụng dấu “;” để ngăn cách các câu lệnh. Phần này bắt buộc phải có.

Chú ý:

- Turbo Pascal cũng sử dụng dấu “;” để kết thúc phần này chuyển qua phần khác cũng như giữa khai báo này qua khai báo khác của chương trình.

- Khi soạn thảo chương trình cho phép đưa vào các câu chú thích nhưng phải được đặt trong cặp dấu móc {...} hoặc (*...*).

CHƯƠNG 2. CÁC KIỂU DỮ LIỆU ĐƠN GIẢN

Mở đầu:

- Chương nầy giới thiệu tổng quan về các kiểu dữ liệu của Pascal và đi sâu mô tả các kiểu dữ liệu đơn giản chuẩn cùng các kiểu dữ liệu do người dùng định nghĩa. Đối với mỗi kiểu dữ liệu giới thiệu cách khai báo, cách sử dụng, miền giá trị cùng các phép toán tác động lên kiểu dữ liệu đó.

- Đưa ra một số thủ tục và hàm chuẩn trên kiểu dữ liệu đơn giản chuẩn và cách sử dụng các thủ tục và hàm chuẩn nầy để viết một biểu thức số học thông thường sang cú pháp của ngôn ngữ Pascal.

Mục tiêu: Học xong chương này Sinh viên

- Biết được tên của một số kiểu dữ liệu chuẩn: nguyên, thực, kí tự, logic biết được giới hạn biểu diễn của mỗi loại kiểu dữ liệu đó.

- Biết cách khai báo biến để dùng trong chương trình trước khi sử dụng.

- Biết cấu trúc chung của khai báo biến trong ngôn ngữ Pascal.

- Biết sử dụng một số thủ tục và hàm trên các kiểu dữ liệu.

2.1. Tổng quan về các kiểu dữ liệu trong Turbo Pascal

Trong Turbo Pascal các kiểu dữ liệu được chia làm 2 loại:

- Các kiểu dữ liệu đơn giản.

- Các kiểu dữ liệu có cấu trúc.

2.1.1. Các kiểu dữ liệu đơn giản gồm:

Kiểu chuẩn:

- Logic.

- Số nguyên.

- Số thực.

- Ký tự.

Kiểu do người dùng định nghĩa:

- Kiểu đoạn con.

- Kiểu liệt kê.

2.1.2. Các kiểu dữ liệu có cấu trúc:

- Mảng.

- Xâu.

- Tập hợp.

- Bản ghi.

- File

- Con trỏ.

2.2. Các kiểu dữ liệu đơn giản chuẩn

2.2.1. Kiểu logic:

- Từ khóa: Boolean.
- Miền trị: Chỉ có 2 giá trị là true và false.
- Các phép toán: AND, OR, NOT, XOR. (Xem bảng dưới đây)
- Phép so sánh:

Bằng nhau: "=".

Nhỏ hơn: "<".

Nhỏ hơn hoặc bằng: "<=".

Lớn hơn: ">".

Lớn hơn hoặc bằng: ">=".

Khác nhau: "<>".

(Quy ước False bé hơn True.)

A	B	A AND B	A OR B	NOT A	A XOR B
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	T
F	F	F	F	T	F

2.2.2. Kiểu số nguyên:

Có 5 kiểu số nguyên:

- Miền giá trị và từ khóa dùng để khai báo của các kiểu đó như sau:

Tên	Min	Max	Yêu cầu bộ nhớ
Shortint	-128	127	1 byte
Integer	-32.768	32.767	2 byte
Longint	-2.147.483.648	2.147.483.647	4 byte
Byte	0	255	1 byte
Word	0	65.535	2 byte

- *Các phép toán:*

Phép cộng: "+"

Phép trừ: "-"

Phép nhân: "*"

Phép DIV: Phép chia hai số nguyên cho nhau cho kết quả cho phần nguyên của thương. Ví dụ 12 div 5 = 2.

Phép MOD: Phép chia hai số nguyên cho nhau cho kết quả cho số dư của phép chia đó. Ví dụ: 27 mod 8 = 3.

- *Các phép so sánh:* Có đầy đủ các phép so sánh tương tự như kiểu logic. Tuy nhiên ở đây ta đã biết quan hệ thứ tự trên tập hợp các số nguyên.

Chú ý

(1). Trên dữ liệu kiểu số nguyên không tồn tại phép chia thực (/).

(2). Thông thường các số nguyên được biểu diễn trong hệ cơ số thập phân. Nếu muốn biểu diễn trong cơ số 16 thì ta thêm dấu "$" phía trước. Ví dụ $A (tức là số 10 trong hệ thập phân).

(3). Với kiểu số nguyên còn có các phép toán AND, OR, NOT, XOR...

2.2.3. Kiểu số thực: Có 5 kiểu số thực.

- Miền giá trị và từ khóa dùng để khai báo của các kiểu đó như sau:

Tên	Min	Max	Yêu cầu bộ nhớ
Real	$2.9*10^{-39}$	$1.7*10^{38}$	6 byte
Single	$1.5*10^{-47}$	$3.4*10^{38}$	4 byte
Double	$5*10^{-324}$	$1.7*10^{380}$	8 byte
Extended	$3.4*10^{-4932}$	$1.1*10^{4932}$	10 byte
Comp	$-9.2*10^{18}$	$9.2*10^{18}$	4 byte

Chú ý:

(1). Miền giá trị của các kiểu dữ liệu (trừ comp) được hiểu là lấy trong các đoạn:

- [-max, -min]
- [min, max]
- số 0.

Nếu một số <-max hoặc > max thì sẽ không biểu diễn được và xem như là tràn số.

Nếu một số >-min nhỏ hơn 0 hoặc lớn hơn 0 nhỏ hơn min thì được xem như là 0.

(2). Chế độ mặc định của TurboPascal là chỉ cho phép làm việc với kiểu số thực REAL. Muốn sử dụng các kiểu khác ta vào bảng chọn OPTION -> COMPILER -> NUMERIC PROCESING rồi đánh dấu [x] vào mục 8087/80287. Khi soạn thảo chương trình ta phải thêm vào chỉ thị dịch {$N+} vào đầu chương trình ngay sau phần tiêu đề.

(3). Trong máy tính số thực được biểu diễn dưới hai dạng:

- Dấu chấm tĩnh, ví dụ: 3.14, 123.456...

- Dấu chấm động (chế độ mặc định), ví dụ: 3.1400000000E+02 (tức là $3.14*10^2$).

- Các phép toán:

Cộng: "+"

Trừ: "-"

Nhân: "*'

Chia: "/"

Chú ý: Với kiểu số thực không tồn tại các phép toán DIV và MOD. Các phép so sánh: Tương tự số nguyên gồm: =, <, >, >= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng), <> (khác nhau).

2.2.4. Kiểu ký tự (CHAR):

- Từ khóa: CHAR.

- Miền giá trị: Các ký tự trong bảng mã ASCII bao gồm:

0..31 : Các ký tự điều khiển.

32.. 127: Các ký tự thông dụng.

128..255: Các ký tự đặc biệt (đồ họa).

Ví dụ: ký tự 'A' có mã là 65; Ký tự 'a' có mã là 97.

Chú ý: Để phân biệt ký tự cũng như xâu ký tự với các đối tượng khác, Pascal quy định khi biểu diễn chúng phải đặt trong cặp dấu nháy đơn, ví dụ 'a', 'abc'.

- Phép so sánh: Muốn so sánh 2 ký tự ta so sánh các mã ASCII tương ứng của chúng, ký tự nào có mã ASCII lớn hơn được xem là lớn hơn. Ví dụ 'a' > 'A' Vì 97 > 65.

Chú ý: Một kiểu dữ liệu được gọi là vô hướng đếm được nếu miền trị của nó là một tập hợp đếm được và trên đó tồn tại quan hệ thứ tự. Ví dụ: Kiểu Byte, kiểu Integer; kiểu Char; kiểu Boolean...

2.3. Kiểu liệt kê và kiểu đoạn con

2.3.1. Kiểu liệt kê:

a. Định nghĩa kiểu và khai báo biến:

- Định nghĩa kiểu:

```
Type ten_kieu = (Danh sách các giá trị);
```

Giải thích:

- ten_kieu ký hiệu cho tên kiểu do người dùng tự đặt.

- Danh sách các giá trị là tập hợp các giá trị được phân cách bởi dấu phẩy ",".

- Khai báo biến:

```
Var  ten_bien : ten_kieu;
```

Giải thích:

- ten_bien ký hiệu cho tên biến do người dùng tự đặt.

- ten_kieu đã được khai báo trước;

Ví dụ:

```
Type  ngay = (sun, mon, tue, wed, thu, fri, sat);
cviec = (dihoc, lambai, thuchanh, nghi);
Var  hqua, hnay: ngay;
Viec: cviec;
```

Chú ý: Cũng có thể khai báo biến kiểu liệt kê trực tiếp mà không cần định nghĩa kiểu.

Ví dụ: Var gioitinh: (nam, nu)

Mau: (xanh, do, tim, vang);

b. Các tác động lên dữ liệu kiểu liệt kê:

- Có thể thực hiện *phép gán* giá trị cho biến.

- *Phép so sánh:* Giá trị liệt kê trong danh sách đi trước được xem là bé hơn giá trị đi sau.

Ví dụ: Với khai báo ở trên thì sun < mon; xanh< do.

c. Chú ý:

(1). Không thể nhập, xuất với dữ liệu kiểu liệt kê.

(2). Giá trị kiểu liệt kê thường được dùng để làm chỉ số cho lệnh lặp for, các trường hợp chọn lựa trong lệnh Case, hoặc chỉ số cho mảng.

2.3.2. Kiểu đoạn con:

a. Định nghĩa kiểu và khai báo biến:

- Định nghĩa kiểu:

```
Type ten_kieu = hang_duoi..hang_tren;
```

Giải thích:

Hang_duoi tức là ký hiệu giá trị hằng cận dưới, hang_tren tức là giá trị hằng cận trên tất nhiên là hang_trên ≥ hang_duoi và phải có cùng kiểu vô hướng đếm được.

- Khai báo biến:

```
Var ten_bien : ten_kieu;          (ten_kieu đã được khai báo)
```

Ví dụ: Type Ky_tu_so: '0'..'9';

Chu_cai: 'a'..'z';

Var a, b: Ky_tu_so;

Ch: chu_cai;

b. Chú ý:

(1). Cũng có thể khai báo biến kiểu đoạn con trực tiếp không thông qua khai báo kiểu.

Ví dụ: Var m, n : 1..100;

Ch: 'a'..'z';

(2). Kiểu đoạn con thường được sử dụng làm tập chỉ số cho biến mảng hay tập hằng trong câu lệnh case.

2.4. Các hàm và thủ tục chuẩn

2.4.1. Các hàm trên kiểu số nguyên và số thực:

Giả sử x là một số nào đó khí đó ta có các hàm sau đây:

ROUND(x) -----→ Cho giá trị là số nguyên gần x nhất

TRUNC(x) ------→ Cho giá trị là phần nguyên của số x } kiểu số nguyên

INT(x) ---------→ Cho giá trị là phần nguyên của số x

FRAC(x) -------→ Cho giá trị là phần thập phân của số x.

ABS(x) --------→ Cho giá trị là giá trị tuyệt đối của số x.

SIN(x) ---------→ Cho giá trị là sinx.

COS(x) - -------→ Cho giá trị là cosx.

EXP(x) --------→ Cho giá trị là e^x.

SQR(x) --------→ Cho giá trị là x^2.

SQRT(x) -------→ Cho giá trị là căn bậc hai của số x (x>=0).

LN(x) ----------→ Cho giá trị là lnx (x>0).

2.4.2. Các hàm khác:

Giả sử var là biến có kiểu vô hướng đếm được.

INC(var, r) ----→ Tăng giá trị của biến var lên r.

DEC(var, r) ----→ Giảm giá trị của biến var đi r (var có kiểu vô hướng đếm được).

Chú ý: Ký hiệu INC(var) tức là INC(var,1) và DEC(var) tức là DEC(var, 1).

PRED(var) ----→ Cho giá trị đi ngay trước var. Ví dụ: pred('c') ='b'.

SUCC(var) ----→ Cho giá trị đi ngay sau var. Ví dụ: succ('c')='d'.

ORD(ch) -------→ Cho giá trị là mã ASCII của ký tự ch (ch kiểu char).

CHR(n) --------→ Cho ký tự có mã ASCII bằng số n (0<=n<=255).

UPCASE(ch) --→ Cho kết quả là chữ ch nhưng là chữ in hoa (ch là chữ cái).

ODD(n) --------→ Cho kết quả TRUE nếu n là số lẻ, bằng FALSE nếu n là số chẵn (n là số nguyên).

RANDOM(n) --→ Cho kết quả là một số ngẫu nhiên lấy trong đoạn [0,n] (n là số nguyên; Chú ý để sử dụng được hàm này thì trước đó phải khởi tạo thủ tục Randomize).

KEYPRESSED --→trả về giá trị true nếu có nhấn phím bất kỳ, ngược lại cho giá trị false (muốn sử dụng phải khai báo unit crt).

2.4.3. Sử dụng hàm để viết các biểu thức bằng cú pháp của Turbo Pascal:

1. $a^b = e^{bln(a)} = exp(b*ln(a))$. (a,b>0).
2. $\log_a b = \log_a e * \log_e b = 1/ln(a)*ln(b)$.
3. $(x^3 + \sin(x^2y))^2 = sqr(sqr(x)*x+sin(sqr(x)*y))$.

BÀI TẬP CHƯƠNG 2.

1. Nếu ch là một biến kiểu char thì các hàm sau đây chr(ord(ch)), ord(chr(ch)) cho kết quả gì? Tại sao?
2. Tìm công thức để đổi một ký tự là chữ hoa thành chữ thường.
3. Dùng các hàm và các phép toán để biểu diễn các công thức sau:
   a. $(\cos(x^3)+ \ln(x^5y))^4$
   b. $(p(p-a)(p-b)(p-c))^{1/2}$
   c. $\log_a(3b^2) + a^{bsinx}$
4. Nếu khai báo m, n, k là các biến kiểu byte, biết rằng m=100, n=200. Hãy cho biết giá trị của k trong các trường hợp sau:
   a. k:= m div 2+n;
   b. k:= 3*m + 2*n;
   c. k:= 1.5*m + n;
5. Hãy so sánh các ký tự và các xâu sau đây:
   a. Chữ 'A' và chữ 'a'.
   b. Chữ 'A' và chữ số '1'.
   c. Ký tự trắng và chữ 'a'.
6. Trong Pascal, nếu một biến chỉ nhận giá trị nguyên trong phạm vi từ 10 đến 25532 thì biến đó có thể khai báo bằng các kiểu dữ liệu nào?
7. Biến P có thể nhận các giá trị 5; 10; 15; 20; 30; 60; 90 và biến X có thể nhận các giá trị 0,1; 0,2; 0,3; 0,4; 0,5. Khai báo nào trong các khai báo sau là đúng?

a. var X, P: byte;

b. var P, X: real;

c. var P: real;
   X: byte;

d. var X: real;
   P: byte;

8. Để tính diện tích S của hình vuông có cạnh A với giá trị nguyên nằm trong phạm vi từ 100 đến 200, cách khai báo S nào dưới đây là đúng và ít tốn bộ nhớ nhất.

a. var S: integer; b. var S: real; c. var S: word;

d. var S: longint; e. var S: boolean;

HƯỚNG DẪN GIẢI BÀI TẬP

1. Chr(ord(ch)) cho k quả là ch, còn ord(chr(ch)) sai cú pháp vì hàm chr có đối là một số kiểu byte.

2. Chr(ord(ch)+ord('a') - ord('A')) (trong đó ch là ký tự chữ hoa cần đổi).

4. a. k=250;

b. Không xác định vì tràn số.

c. Không xác định vì sai kiểu.

5. Xem bảng mã ASCII và cách so sánh các ký tự.

CHƯƠNG 3. BIỂU THỨC VÀ CÂU LỆNH ĐƠN GIẢN

Mở đầu:

- Chương nầy đưa ra các kiến thức về các thành phần tạo thành một biểu thức trong Pascal, cách tác động của các toán tử lên các toán hạng trong một biểu thức.

- Giới thiệu các loại câu lệnh đơn giản, cách khai báo và sử dụng các câu lệnh này trong chương trình. Đồng thời mô tả cách sử dụng các câu lệnh này bằng các ví dụ minh họa.

Mục tiêu: Học xong chương này Sinh viên

- Biết được những khái niệm về phép toán, biểu thức số học, hàm số học chuẩn, biểu thức quan hệ.

- Nắm được các phép toán thông dụng trong ngôn ngữ lập trình.

- Biết dùng các câu lệnh và các phát biểu có thể dùng trong thân của chương trình.

- Biết các lệnh nhập dữ liệu từ bàn phím, lệnh xuất dữ liệu ra màn hình máy tính và máy in.

3.1. Biểu thức

3.1.1. Hằng:

Hằng là đại lượng không thay đổi trong chương trình. Có các loại hằng: hằng số, hằng ký tự, hằng xâu, hằng logic... Từ khóa để khai báo hằng là CONST.

Ví dụ: Const max = 100; ch='A'; hoten='ng van A';

3.1.2. Biến :

Biến là đại lượng có thể thay đổi giá trị vào từng thời điểm khác nhau của chương trình. Biến dùng để lưu trữ dữ liệu, biến được khai báo sau từ khóa VAR. Việc khai báo các biến là nhằm cung cấp các vùng nhớ để lưu trữ dữ liệu trong qúa trình thực hiện chương trình.

Ví dụ:

```
Var     a,b,c: Integer;
        Ch : Char;
        Ok : Boolean;
```

Chú ý: Khi khai báo có nhiều biến cùng kiểu thì ta dùng dấu phẩy (",") để ngăn cách. Hết khai báo kiểu dữ liệu này chuyển qua khai báo kiểu dữ liệu khác ta dùng dấu ";"

3.1.3. Toán tử:

Là các phép toán tác động lên dữ liệu (hay còn gọi là toán hạng).

Ví dụ: Các phép +, -, *, /, and, or, not...

Chú ý : Thứ tự ưu tiên thực hiên các phép toán như sau:

- Các phép toán 1 ngôi.
- Các phép *, /, DIV, MOD, AND.
- Các phép +, -, OR, XOR
- Các phép toán so sánh =, <>, <=, >=, <, >.

3.1.4. Toán hạng:

Là một trong các đại lượng sau:

- Hằng
- Biến
- Hàm
- Biểu thức

3.1.5. Biểu thức:

Là một tập hợp gồm các toán tử và các toán hạng trong đó toán tử tác động phù hợp lên toán hạng.

Ví dụ: Biểu thức r*pi + abs(x+y) - 10

Các toán tử : *, +, -

Các toán hạng: r, x, y là các biến; pi, 10 là hằng; abs là hàm.

Chú ý:

(1). Trong một biểu thức các toán tử trong dấu ngoặc "(...)" được ưu tiên thực hiện trước.

(2). Nếu có nhiều phép toán cùng cấp ưu tiên thì các phép toán được thực hiện tuần tự từ trái qua phải (trừ các phép toán tác động lên kiểu logic).

(3). Một biểu thức cho kết quả bằng số được gọi là biểu thức số học, cho kết quả kiểu logic thì được gọi là biểu thức logic.

Ví dụ: Tính giá trị của biểu thức sau:

((18 mod 4 div 2 < 3) >= false) = true
((2 div 2 < 3) >= false) = true
< 3) >= false) = true
(true >= false) = true
true = true ------→ true

3.2. Câu lệnh

3.2.1. Phân loại câu lệnh:

Trong Pascal có hai loại câu lệnh đó là câu lệnh đơn giản và câu lệnh có cấu trúc.

- Câu lệnh đơn giản gồm: Lệnh gán, lệnh xuất, nhập dữ liệu, lệnh nhảy và lời gọi chương trình con dạng thủ tục.

- Câu lệnh có cấu trúc gồm: Lệnh ghép, lệnh rẽ nhánh, lệnh lựa chọn, các lệnh lặp và lệnh WITH.

3.2.2. Câu lệnh đơn giản:

a. Lệnh gán:

Cú pháp: `ten_bien := bt;`

Giải thích bt là ký hiệu cho biểu thức.

Ý nghĩa: Lệnh sẽ thực hiện việc gán giá trị của bt cho biến ten_bien.

Chú ý:

(1). Biến ten_bien và giá trị của bt phải cùng kiểu dữ liệu.

(2). Vế trái luôn là biến còn vế phải là biểu thức, đảo ngược lại là sai cú pháp.

Ví dụ: a:= 5; S : = s + d*i; i:=i+1;

b. Lệnh ghi dữ liệu lên màn hình:

Cú pháp:

```
(1)  WRITE(bt1, bt2,..., btn);       (n≥1)
(2)  WRITELN(bt1, bt2,..., btn);
(3)  WRITELN    ;
```

Giải thích bt1,bt2,...,btn ký hiệu cho biểu thức1, biểu thức2,..., biểu thứcn.

Ý nghĩa:

(1): Lần lượt ghi lên màn hình giá trị của các biểu thức tại vị trí con trỏ.

(2): Tương tự dạng (a) chỉ khác ở chỗ sau khi ghi xong con trỏ được đưa về đầu dòng tiếp theo.

(3): Không ghi gì lên màn hình cả, chỉ làm thao tác đưa con trỏ về đầu dòng tiếp theo.

Ví dụ:

```
Var a,b : integer;
Begin
        a:=5;  b:= 10;
        Writeln('tich cua', a, ' va', b, ' la : ', a*b);
End.
```

Chú ý:

(1). Giá trị của các biểu thức trong câu lệnh write, writeln phải có kiểu chuẩn.

(2). Có thể sử dụng cách ghi dữ liệu có quy cách như sau:

** Với dữ liệu kiểu số nguyên, ký tự, xâu, logic:*

```
(a)  WRITE(bt : n,...);
(b)  WRITELN(bt : n,...);
```

Khi đó giá trị của biểu thức sẽ được ghi lên màn hình theo chế độ căn phải, trong đó n là số vị trí (cột) giành để ghi dữ liệu.

Ví dụ:

WRITE(123 : 5); -------→ ␣␣123

WRITE('abcd' : 8); -------→ ␣␣␣␣abcd

WRITE(3=5:9); -------→ ␣␣␣␣␣False

* *Với dữ liệu kiểu số thực:*

Chế độ mặc định của Pascal là ghi theo dạng dấu chấm động.

Muốn ghi số thực theo dạng dấu chấm tĩnh ta dùng cách ghi sau đây:

```
(a)  WRITE(bt : n : m,...);          (n>m)
(b)  WRITELN(bt : n : m,...);
```

Khi đó giá trị của biểu thức sẽ được ghi lên màn hình theo chế độ căn phải, trong đó n là số vị trí (cột) giành để ghi dữ liệu, m là số chữ số thập phân cần lấy.

Ví dụ: WRITE(123.456 : 12 : 2); -----→ ␣␣␣␣␣123.47

Chú ý: Khi ghi dữ liệu là số thực lên màn hình theo dạng có quy cách có thể số được ghi lên đã được quy tròn (ví dụ trên) nhưng giá trị thực của nó vẫn được lưu giữ trong bộ nhớ.

c. Lệnh nhập dữ liệu từ bàn phím:

Cú pháp:

```
(1)  READ(bien1, bien2,..., bienn);   (các biến đã được khai báo).
(2)  READLN(bien1, bien2,..., bienn);
(3)  READLN;
```

Giải thích bien1,bien2,...,bienn ký hiệu cho biến1, biến2,...,biếnn.

Ý nghĩa:

(1): Cho phép lần lượt nhập dữ liệu cho các biến.

(2): Tương tự dạng (a) chỉ khác ở la khi nhập xong con trỏ được đưa về đầu dòng tiếp theo.

(3): Chỉ làm thao tác tạm dừng chương trình, chờ gõ phím Enter để tiếp tục.

Chú ý:

(1). Nguyên tắc nhập dữ liệu là trong câu lệnh có bao nhiêu biến thì ta phải nhập đủ giá trị cho bấy nhiêu biến. Giữa giá trị của biến này và biến kia được ngăn cách bởi ít nhất là một ký tự trắng; kết thúc việc nhập bằng cách nhấn phím ENTER

(Nếu chưa nhập đủ giá trị cho các biến thì chương trình vẫn tiếp tục dừng lại chờ nhập tiếp, chừng nào nhập đủ mới tiếp tục, Ví dụ Khi gặp lệnh READ(a,b,c); chương trình tạm dừng chờ nhập dữ liệu và khi đó ta nhập chẳng hạn 3 ␣ 9 ␣ 12 ↵)

(2). Thông thường khi thực hiện chương trình để nhập dữ liệu ta nên ghi lên màn hình câu hướng dẫn cho người thực hiện chương trình, nghĩa là trước câu lệnh READ, READLN nên có một câu lệnh WRITE.

Ví dụ: Write(' nhập n nguyên duong : '); readln(n);

(3). Câu lệnh READLN thường được sử dụng tạm dừng chương trình (màn hình) để xem kết quả, vì vậy thường được sử dụng ở cuối của một chương trình (trước từ khóa "END.")

(4). Với dữ liệu kiểu logic không cho phép nhập từ bàn phím.

Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào chiều dài và chiều rộng của một hình chữ nhật rồi in lên màn hình chu vi và diện tích của hình chữ nhật đó.

Hướng dẫn:

- Khai báo: Các biến a,b kiểu số thực đề lưu chiều dài, chiều rộng.

  Biến p để chứa chu vi, biến s để chứa diện tích.

- Nhập dữ liệu cho a,b.

- Tính chu vi theo công thức p:=2*(a+b);

- Tính diện tích theo công thức s:= a*b;

- Ghi dữ liệu lên màn hình theo dạng có quy cách (số thực).

Chương trình:

```
Program  hinh_chu_nhat;
Var a,b,p,s : real;
Begin
     Write('nhap chieu dai a: '); readln(a);
     Write('nhap chieu rong b: '); readln(b);
     p:=2*(a+b);  s:= a*b;
     Writeln('chu vi cua hinh chu nhat la: ',p:10:2);
     Writeln('Dien tich cua hinh chu nhat la: ',s:10:2);
     Readln
End.
```

Ví dụ 2: Viết chương trình nhập vào một số nguyên dương có ba chữ số rồi in lên màn hình số đảo ngược của số đó.

Hướng dẫn:

- Khai báo: Biến n để chứa dữ liệu mhập vào.

- Nhập n nguyên dương có 3 chữ số.

- Dùng các phép toán MOD và DIV lần lượt tách các chữ số hàng đơn vị, hàng trăm, hàng nghìn và ghi lên màn hình.

Chương trình:

```
Program  So_dao;
Var n : integer ;
Begin
       Write('nhap so nguyen co 3 chu so : '); readln(n);
       Write('so dao nguoc cua so ',n,' la: ');
       writeln(n mod 10,n mod 100 div 10,n div 100);
       Readln
End.
```

Ví dụ 3: Viết chương trình tính tổng:

S = 1+2+3+...+n (n nhập từ bàn phím).

Hướng dẫn:

- Khai báo: Biến n kiểu số nguyên để chứa dữ liệu nhập vào.
  Biến S kiểu số nguyên để lưu kết quả.
- Nhập dữ liệu cho n.
- Khởi gán s:=0;
- Tính S theo công thức S=n*(n+1) div 2;
- Ghi dữ liệu lên màn hình.

Chương trình:

```
Program  Tinh_tong;
Var n,s : integer ;
Begin
     Write('nhap so nguyen duong n: '); readln(n);
     s:=0;
     s:=n*(n+1) div 2;
     Writeln('Tong', n, ' so tu nhien dau tien la: ',s);
     Readln
End.
```

BÀI TẬP CHƯƠNG 3.

1. Viết chương trình nhập độ dài bán kính của một hình tròn, in lên màn hình chu vi và diện tích của hình tròn đó.

2. a. Viết chương trình nhập một ký tự bất kỳ sau đó in lên màn hình mã ASCII của ký tự đó.

b. Viết chương trình nhập một số nguyên trong đoạn [32..127], in lên màn hình ký tự có mã ASCII là số đó.

3. a. Viết chương trình nhập vào số nguyên dương n, số thực a rồi in lên màn hình giá trị căn bậc n của a.

b. Viết chương trình nhập các số dương a,b rồi in lên màn hình $\log_a b$.

4. Viết chương trình nhập vào 3 cột điểm với các hệ số 1,2 và 3. Tính và in lên màn hình điểm trung bình cộng.

5. Viết chương trình nhập vào số nguyên dương n, in lên màn hình tổng sau:

$S= 1^2 + 2^2 + 3^2 + ... + n^2$.

(Biết công thức s=n*(n+1)*(2*n+1) div 6)

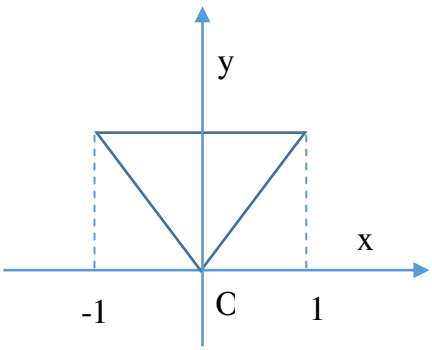
6. Hãy viết biểu thức toán học dưới đây trong Pascal:

$$(1+z)\frac{x+\frac{y}{z}}{a-\frac{1}{1+x^3}}$$

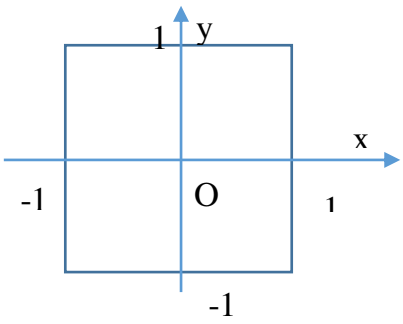
7. Hãy chuyển các biểu thức trong Pascal dưới đây sang biểu thức toán học tương ứng.

a. a/b*2; b. a*b*c/2; c. 1/a*b/c; d. b/sqrt(a*a+b);

8. Hãy viết biểu thức lôgic cho kết quả true khi tọa độ (x, y) là điểm nằm trong vùng gạch chéo kể cả biên của các hình 2.a và 2.b.

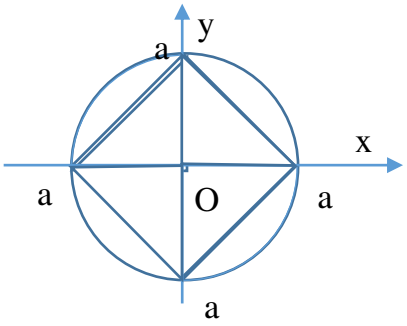


Hình 2a.



Hình 2b.

9. Hãy viết chương trình nhập số a (a>0) rồi tính và đưa ra diện tích phần được gạch chéo trong hình 3 (kết quả làm tròn đến bốn chữ số thập phân).



Hình 3

10. Lập trình và đưa ra màn hình vận tốc v khi chạm đất của một vật rơi từ độ cao h, biết rằng $v = \sqrt{2gh}$, trong đó g là gia tốc rơi tự do và g = 9,8m^2. Độ cao h (m) được nhập vào từ bán phím.

HƯỚNG DẪN GIẢI BÀI TẬP

1. - Khai báo biến r kiểu số thực để nhập bán kính, các biến p, s kiểu số thực để chứa chu vi và diện tích.

- Nhập bán kính r.
- Tính chu vi và diện tích theo công thức.
- Thông báo kết quả.

2a - Khai báo biến ch kiểu char để nhập ký tự bất kỳ.

- Nhập ký tự.
- Dùng hàm ORD(ch) để tìm mã ASCII của ký tự đó và thông báo kết quả.

2b - Khai báo biến n kiểu số nguyên để nhập dữ liệu.

- Nhập số nguyên n.
- Dùng hàm CHR(n) để tìm ký tự tương ứng với mã ASCII của số n đó và thông báo kết quả.

3a. - Khai báo biến n kiểu số nguyên, a kiểu số thực để nhập dữ liệu.

- Nhập n,a.
- Dùng công thức exp(1/n*ln(a)) để tính và thông báo kết quả.

3b. - Khai báo các biến a,b kiểu số thực để nhập dữ liệu.

- Nhập a,b.
- Dùng công thức exp(b*ln(a)) để tính và thông báo kết quả.

4. - Khai báo các biến a,b,c kiểu số thực để nhập 3 cột điểm; biến tbc kiểu số thực để chứa kết quả.

- Nhập a, b, c.
- Dùng công thức tbc:= (a+2*b+3*c)/6
- Thông báo kết quả.

5. - Khai báo biến n, s kiểu số nguyên để nhập dữ liệu và chứa kết quả.

- Nhập n (nguyên dương).
- Tính s theo công thức hướng dẫn. - Thông báo kết quả.

CHƯƠNG 4. CÂU LỆNH CÓ CẤU TRÚC

Mở đầu:

Nội dung chương nầy đưa ra các câu lệnh có cấu trúc (If, Case, For, Repeat, While). Trong mỗi lệnh đưa ra cú pháp của từng lệnh và ví dụ minh họa cho từng cú pháp lệnh. Đồng thời phân tích cách sử dụng các câu lệnh như thế nào cho phù hợp.

Mục tiêu: Học xong chương này Sinh viên

- Hiểu khái niệm rẽ nhánh, biết được cấu trúc chung của cấu trúc rẽ nhánh.
- Biết sử dụng các câu lệnh rẽ nhánh cho phù hợp.
- Hiểu được ý nghĩa và tác dụng của cấu trúc lặp.
- Biết dùng các câu lệnh lặp và cách dùng vòng lặp nào cho phù hợp với yêu cầu bài toán.

4.1. Lệnh ghép

4.1.1. Khái niệm: Lệnh ghép là tập hợp các lệnh đặt giữa cặp từ khóa begin....end;

4.1.2. Cú pháp:

```
Begin
     CL1;
     CL2;
     ...
end;
```

(Trong đó CL1, CL2... ký hiệu cho câu lệnh1, câu lênh2,...)

4.1.3. Ý nghĩa: *Ghép nhiều lệnh thành một câu lệnh.*

4.2. Lệnh IF (câu lệnh rẽ nhánh).

4.2.1. Cú pháp: Có 2 dạng:

```
(1)   If   <BTLG>   then CL1 else CL2;
(2)   If   <BTLG>   then CL;
```

Trong đó BTLG ký hiệu cho biểu thức logic.

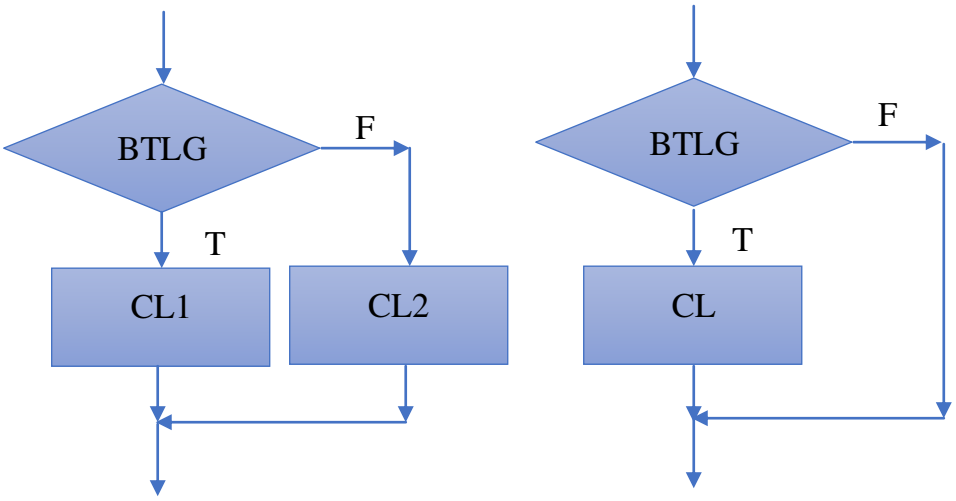
Dạng (1) còn được gọi là dạng đầy đủ, Dạng (2) được gọi là dạng thiếu.

4.2.2. Ý nghĩa:

(1) Nếu BTLG nhận giá trị true thì máy sẽ thực hiện CL1, ngược lại máy sẽ thực hiện CL2.

(2) Nếu BTLG nhận giá trị true thì máy sẽ thực hiện CL, ngược lại thì không thực hiện.

4.2.3. Sơ đồ hoạt động:



Dạng (1) *Dạng (2)*

4.2.4. Chú ý:

Với lệnh If dạng (1) sau CL1 (trước else) không có dấu ";" vì nếu có thì máy sẽ hiểu là lệnh if dạng (2) sau đó gặp từ khóa else máy sẽ báo lỗi vì không có lệnh này.

4.2.5. Các ví dụ:

Ví dụ 1: Viết chương trình giải phương trình ax+b=0.

Hướng dẫn:

- Khai báo các biến a, b kiểu số thực để nhập dữ liệu, x để chứa nghiệm.
- Nhập a,b.
- Biện luận nghiệm theo a và b.
- Thông báo kết quả lên màn hình cho từng trường hợp.

Chương trình:

```
Program  ph_tr_bac_nhat;
Var a,b,x : real;
Begin
     Write('nhap he so a,b : '); readln(a,b);
     If a<>0 then
       Begin  x:= -b/a;
          Writeln('nghiem duy nhat la: ',x:5:2);
       End
     Else
       If b<>0 then Writeln('phuong trinh vo nghiem')
       Else Writeln('pt vo dinh - nghiem la moi x ');
  Readln
End.
```

Ví dụ 2: Viết chương trình tính giá trị của hàm:

$$f(x) = \begin{cases} \dfrac{|x^2+1|}{\sqrt{x}} & x>0 \\ \sqrt[3]{x^2-1} & 0\geq x \end{cases}$$

Hướng dẫn:

- Khai báo các biến x kiểu số thực để nhập đối số, f để chứa giá trị hàm.
- Nhập x.
- Biện luận theo x để tính f.
- Thông báo kết quả lên màn hình số thực f theo dạng có quy cách.

Chương trình:

```
Program  Ham_so;
Var x,f : real;
Begin
     Write('nhap he doi so x : '); readln(x);
     If x>0 then f:=abs(sqr(x)+1)
     Else f:= exp(1/3*ln(sqr(x)-1);
     Writeln(' f(',x:5:2,')=',f:8:2)
     Readln
End.
```

4.3. Lệnh CASE (câu lệnh lựa chọn)

***4.3.1. Cú pháp*:** Có 2 dạng:

```
(1)  Case < bt > of
          TH1: CL1;
          TH2: CL2;
          ...
          THn: CLn;
     Else  CLn+1;
     End;
(2)  Case < bt > of
          TH1: CL1;
          TH2: CL2;
          ...
          THn: CLn;
     End;
```

4.3.2. Giải thích:

bt ký hiệu cho biểu thức; TH1,TH2,...,THn ký hiệu cho các tập hằng; CL1, CL2, ..., CLn, CLn+1 ký hiệu cho các câu lệnh.

4.3.3. Ý nghĩa:

(1) Tùy thuộc vào giá trị của biểu thức bt thuộc vào tập hằng nào mà máy tính sẽ thực hiện câu lệnh tương ứng. Nếu không sẽ thực hiện CLn+1 ;

(2) Tương tự dạng (1), chỉ khác ở chổ nếu giá trị của biểu thức không thuộc tập hằng nào cả thì máy sẽ không thực hiện câu lệnh nào cả trong thân lệnh CASE và sẽ thoát.

4.3.4. Chú ý:

(1). Giá trị của bt phải có cùng kiểu với các phần tử của tập hằng và phải là kiểu vô hướng đếm được.

(2). Tập hằng có thể là một giá trị, một tập hợp các giá trị, và thậm chí là một kiểu dữ liệu (kiểu đoạn con, liệt kê).

(3). Lệnh CASE có thể được thay thế bởi nhiều lệnh IF lồng nhau và ngược lại lệnh IF cũng có thể được thay thế bằng lệnh CASE ví dụ lệnh IF dạng (1) có thể được thay thế bằng:

```
CASE <BTLG>  OF
    True:  CL1;
    False: CL2;
END;
```

Tuy nhiên sử dụng lệnh case hay if là tùy vào từng bài toán cụ thể.

4.3.5. Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào một số n bất kỳ trong đoạn [0...255] và thông báo lên màn hình:

- Câu thông báo: ' Day la ky tu dieu khien'. (nếu n < 32);

- Câu thông báo: ' la ky tu thong dung'. (nếu 127 ≥ n ≥ 32);

- Câu thông báo: ' la ky tu dac biet '. (nếu n ≥ 128);

Hướng dẫn:

- Khai báo biến n kiểu byte để nhập dữ liệu.

- Nhập n.

- Dùng lệnh CASE để phân chia các trường hợp theo n và thông báo kết quả lên màn hình theo yêu cầu.

Chương trình:

```
Var n : byte;
Begin
   Write('nhap so n (0<=n<=255) : ');
   Readln(n);
   CASE n of
     0..31    : Writeln(' Day la ky tu dieu khien');
     32..127  : Writeln(chr(n),'la ky tu thong dung');
     128..255 : Writeln(chr(n),' la ky tu dac biet');
   END;
   Readln
```

```
End.
```

Ví dụ 2: Viết chương trình nhập vào tháng t ($12 \geq t \geq 1$), có thể nhập thêm năm sau đó in lên màn hình số ngày của tháng đó.

Hướng dẫn:

- Khai báo biến t kiểu byte để nhập tháng; biến n kiểu số nguyên để nhập năm; biến sn kiểu byte để chứa số ngày của tháng.
- Nhập t.
- Dùng lệnh CASE để phân chia các trường hợp theo t:

* t∈[1,3,5,7,8,10,12] thì sn:=31;

* t∈[2,4,6,9,11] thì sn:=30;

* t=2 thì nhập thêm năm n, nếu n là năm nhuận thì sn:=29, ngược lại sn:=28;

- Thông báo kết quả sn lên màn hình.

Chương trình:

```
Var t,sn : byte;
 n: Integer;
Begin
     Write('nhap so thang t (1<=t<=12) : ');
     readln(t);
  CASE  t  of
     1,3,5,7,8,10,12:  sn:=31;
     4,6,9,11       :  sn:=30;
     2              : Begin
                         Write(' nhap nam: ');
                         readln(n);
                         If  n  mod  4 = 0 then sn:=29
                         Else sn:=28;
                      End;
     ELSE  writeln('khong  co thang do');
  END;
 If t<=12 then Writeln('so ngay cua thang ',t,'la: ',sn);
 Readln
End.
```

4.4. Lệnh FOR (Lặp biết trước số lần).

4.4.1. Cú pháp: Có 2 dạng:

```
(1)  For  bdk:= bt1 to bt2    do CL;
(2)  For  bdk:= bt2 downto bt1 do CL;
```

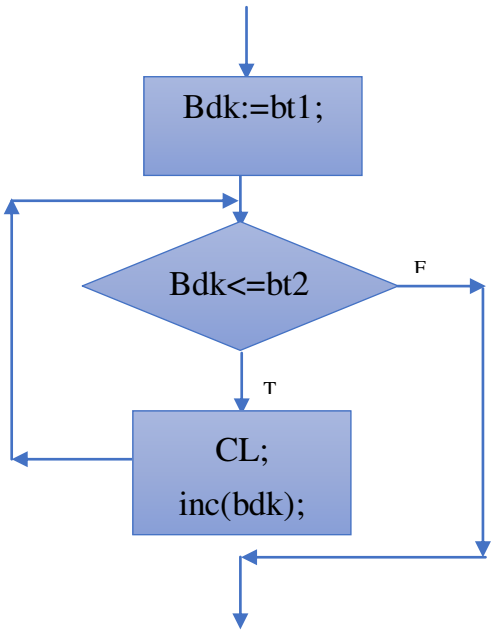
(Trong đó bdk ký hiệu cho biến điều khiển; bt ký hiệu cho biểu thức).

4.4.2. Ý nghĩa:

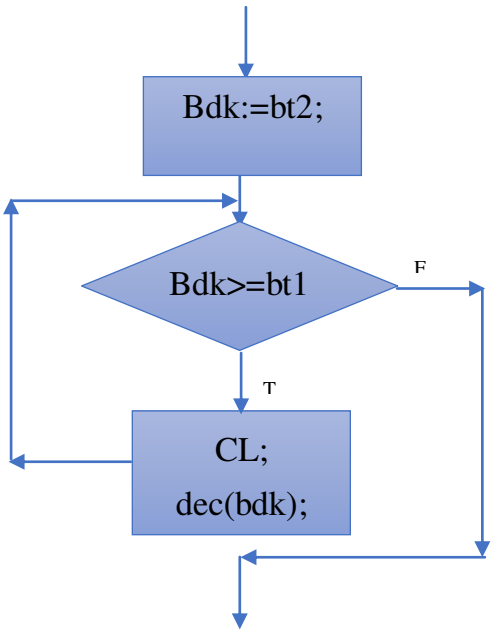
(1) Câu lệnh CL sẽ được lặp đúng bằng số lần mà bdk được gán tăng dần lần lượt từ giá trị của bt1 đến giá trị của bt2.

(2) Câu_lệnh sẽ được lặp đúng bằng số lần mà bdk được gán giảm dần lần lượt từ giá trị của bt2 đến giá trị của bt1.

4.4.3. Sơ đồ hoạt động:



Dạng (1)



Dạng (2)

4.4.4. Chú ý:

(1). Biến điều khiển bdk phải có kiểu vô hướng đếm được (thường là kiểu số nguyên).

(2). Trong thân câu lệnh for không được làm thay đổi giá trị của bdk vì như thế sẽ không kiểm soát được biến điều khiển và có thể dẫn đến trình trạng lặp vô hạn. Khi đó ta sử dụng tổ hợp phím Ctrl + Break để dừng chương trình.

(3). Có thể sử dụng lệnh BREAK để thoát khỏi lệnh FOR cũng như thoát khỏi các lệnh WHILE, REPEAT.

4.4.5. Các ví dụ:

Ví dụ 1: Viết chương trình hiển thị lên màn hình mã ASCII và các ký tự tương ứng, cứ 20 ký tự trên một trang màn hình.

Hướng dẫn:

- Khai báo biến điều khiển lệnh FOR là i kiểu byte;
- Dùng lệnh FOR dạng (1) với biến điều khiển i chạy từ 0 đến 255, với mỗi giá trị của i ta ghi lên màn hình ký tự có mã ASCII tương ứng; Sau đó kiểm tra xem nếu i+1 chia hết cho 20 thì dùng lệnh readln để tạm dừng màn hình.

Chương trình:

```
Var i : byte;
Begin
   For i:=0 to 255 do
     Begin
        Writeln('ky tu tuong ung voi ma',i,' la:', chr(i));
        If  i+1 mod 20 = 0 then readln;
     End;
End.
```

Ví dụ 2: Viết chương trình tính tổng: $S = 1^3+2^3+3^3+...+n^3$ (n nhập từ bàn phím)

Hướng dẫn:

- Khai báo: Biến n kiểu số nguyên integer để chứa dữ liệu nhập vào.

  Biến S kiểu longint để lưu kết qua.

- Nhập n.

- Khởi gán S:=0;

- Dùng lệnh FOR dạng (1) với biến điều khiển i chạy từ 1 đến n, với mỗi giá trị của i ta thực hiện lệnh cộng dồn S:= S+ sqr(i)*i;

- Ghi kết quả S lên màn hình.

Chương trình:

```
Program  Tinh_tong;
Var n,s,i : integer ;
Begin
      Write('nhap so nguyen duong n: '); readln(n);
      s:=0;
      For i:=1 to n do    s:=s+sqr(i)*i;
      Write('Tong', n, ' so tu nhien dau tien la: ',s);
      Readln
End.
```

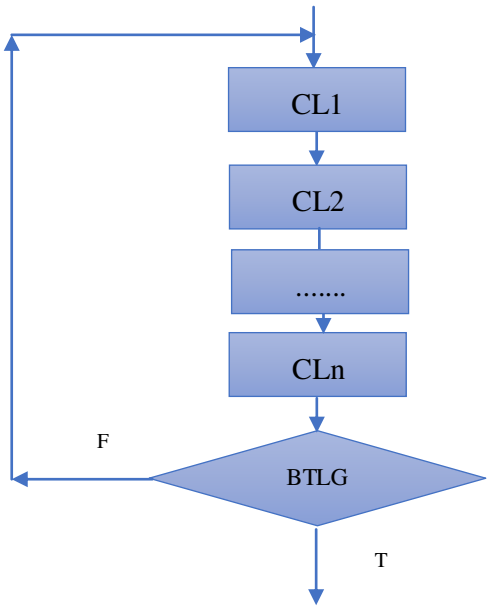
4.5. Lệnh REPEAT (lặp với số lần không biết trước).

4.5.1. Cú pháp:

```
Repeat
     CL1;
     CL2;
     ...
     CLn;
Until <BTLG>;
```

4.5.2. Ý nghĩa: Thực hiện lần lượt các câu lệnh từ CL1 đến CLn sau đó kiểm tra giá trị của biểu thức logic, nếu sai thì lặp lại, nếu đúng thì thoát.

4.5.3. Sơ đồ hoạt động:



4.5.4. Chú ý:

(1). Các câu lệnh nằm giữa repeat... until không cần phải sử dụng cặp từ khóa begin...end;

(2). Trong thân của lệnh repeat thường phải có câu lệnh làm thay đổi giá trị của biểu thức logic nếu không dễ dẫn đến tình trạng lặp vô hạn.

(3). Khi thực hiện chương trình nếu bị lặp vô hạn thì nhấn tổ hợp phím CTRL-BREAK để dừng chương trình.

(4). Lệnh repeat thường được sử dụng khi phải thực hiện ít nhất là một lần các câu lệnh từ CL1 đến CLn và số lần lặp không xác định được trước. (Trong trường hợp biết trước số lần lặp ta nên dùng lệnh for).

4.5.5. Các ví dụ:

Ví dụ 1: Sử dụng lệnh repeat để lọc dữ liệu vào theo yêu cầu, chẳng hạn cần nhập n nguyên dương ta viết như sau:

```
Repeat
      Write('nhap n >0 : ');
      Readln(n);
Until n>0;
```

Ví dụ 2: Viết chương trình nhập điểm 6 môn thi sau đó tính điểm trung bình, với yêu cầu mỗi lần tính xong lại thông báo lên màn hình câu: ' Tiep tuc hay khong (c/k)?', nếu nhập 'c' thì lại tiếp tục, nếu nhập 'k' thì kết thúc.

```
Program  Diem_trung_binh;
Var a,b,c,d,e,f : real;
      Ch:char;
Begin
```

```
        Repeat
             Write('nhap diem 6 mon thi a,b,c,d,e,f : ');
             readln(a,b,c,d,e,f);
             Writeln('DTB la:',(a+b+c+d+e+f)/6:4:2);
             Writeln('Tiep tuc nua khong?'); readln(ch);
        Until upcase(ch)='K';
End.
```

Ví dụ 3: Viết chương trình nhập vào một số nguyên dương bất kỳ rồi in lên màn hình số chữ số của số đó.

Hướng dẫn:

-Khai báo biến n và biến dem kiểu Integer để nhập dữ liệu và đếm số chữ số.

-Nhập n.

-Khởi gán biến dem:=0.

-Mỗi lần tăng biến dem lên 1 thì loại bỏ đi chữ số sau cùng của n bằng lệnh gán lại n:=n div 10 và lại lặp lại cho đến khi n = 0 thì thôi.

-Thông báo kết quả là giá trị biến dem lên màn hình.

Chương trình:

```
Program  Dem_so_chu_so;
Var n,dem : integer;
Begin
     Repeat
          Write('nhap n >0 : ');Readln(n);
     Until n>0;
     Dem:=0; n1 := n;
     Repeat
          Dem:=dem+1;
          n:=n div 10;
     until n = 0;
     writeln('so', n1, ' co ',dem,' chu so');
     readln
End.
```

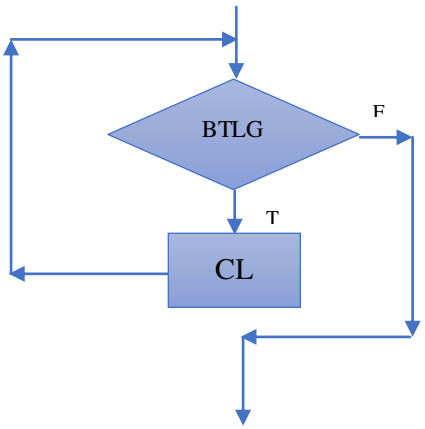
4.6. Lệnh WHILE (lặp với số lần không biết trước).

4.6.1. Cú pháp:

```
While <BTLG> do CL;
```

4.6.2. Ý nghĩa: Khi nào BTLG còn nhận giá trị true thì máy còn thực hiện câu lệnh CL.

4.6.3. Sơ đồ hoạt động:



4.6.4. Chú ý:

(1). Cũng giống lệnh Repeat, trong thân của lệnh while phải có câu lệnh làm thay đổi giá trị của biểu thức logic nếu không dễ dẫn đến tình trạng lặp vô hạn.

(2). Khác với lệnh repeat lệnh while kiểm tra điều kiện của biểu thức logic trước, nên có thể câu lệnh trong thân của nó không được thực hiện một lần nào cả khi mà ngay từ đầu biểu thức logic nhận giá trị sai.

4.6.5. Các ví dụ:

Ví dụ 1: Sử dụng lệnh while, viết chương trình nhập vào một số nguyên dương bất kỳ rồi in lên màn hình số chữ số của số đó.

Chương trình:

```
Program  Dem_so_chu_so;
Var n,dem : integer;
Begin
     Repeat
          Write('nhap n >0 : ');
          Readln(n);
     Until n>0;
     Dem:=0; n1 := n;
     While n>0 do
     Begin
                    Dem:=dem+1;
                    n:=n div 10;
                End;
     writeln('so', n1, ' co ',dem,' chu so');
     readln
End.
```

Ví dụ 2: Viết chương trình nhập vào hai số nguyên dương rồi tìm ước số chung lớn nhất của chúng.

```
Program  UCLN;
Var m,n: integer;
Begin
```

```
        Repeat
             Write('nhap m,n nguyen duong : ');
             Readln(m,n);
        Until (m>0) and (n>0);
        While m <>n  do
        If  m>n then m:=m-n
                    else n:=n-m;
        writeln('UCLN cua hai so do la: ',m);
        readln
End.
```

BÀI TẬP CHƯƠNG 4.

4.1. Lệnh If và Case:

1. Viết chương trình nhập vào 3 số nguyên, in ra màn hình số lớn nhất.

2. Viết chương trình giải phương trình bậc hai.

3. Viết chương trình nhập 3 số, cho biết chúng có thể là độ dài 3 cạnh của một tam giác hay không?

4. Viết chương trình nhập vào số KW điện tiêu thụ của một hộ gia đình, in ra màn hình số tiền tương ứng, biết rằng giá thành được tính như sau:

- Từ KW thứ 1 đến thứ 100 giá 0.5 nghìn đồng.
- Từ KW thứ 101 đến thứ 200 giá 0.8 nghìn đồng.
- Từ KW thứ 201 trở lên giá 1.5 nghìn đồng.

5. Đoạn chương trình sau đây đúng cú pháp hay không? Nếu đúng cho biết ý nghĩa của nó:

```
Var a,b : Integer;
Begin
     Write(' nhap a,b: '); readln(a,b);
     Case  a<b  of
          True:  writeln(a, '<', b);
          False: writeln(b, '<=', a);
     End;
     Readln
End.
```

4.2. Lệnh For, Repeat và While:

1. Viết chương trình tính các tổng sau:

a. $S = 1 - 2 + 3 - ... + (-1)^{n-1}n$

b. $S = 1 + 1*2 + 1*2*3 + ... + 1*2*3*...*n$

c. $S = 1 - 1*2^2 + 1*2*3^2 - ... + (-1)^{n-1}1*2*3*...*n^2$

2. Viết chương trình tìm bội số chung nhỏ nhất của hai số nguyên dương.

3. Viết chương trình nhập vào một số nguyên dương, in ra màn hình số đảo ngược của số đó.

4. Viết chương trình nhập vào hai số nguyên dương m,n (m<n) cho biết tất cả các số hoàn hảo nằm trong đoạn [m,n].

(Số hoàn hảo là số bằng tổng tất cả các ước của nó (trừ số đó) chẳng hạn 6, 28,...).

5. Viết chương trình kiểm tra xem một số nguyên n nào đó có phải là số nguyên tố hay không?

6. Viết chương trình tìm tất cả các số nguyên tố nằm trong đoạn [1,n] (n nhập từ bàn phím).

7. Tìm số của dãy FIBONACCI lớn nhất thỏa mãn bé hơn 2000. (Dãy Fibonacci là dãy $a_1=1$, $a_2=1$, $a_n=a_{n-1}+a_{n-2}$ (n>=3)).

8. Một người có số tiền t đem gửi ngân hàng với lãi suất k/tháng (biết rằng lãi của tháng trước được nhập vào tiền gốc để tính lãi cho tháng tiếp theo). Hỏi để có được số tiền max (max>t) thì người đó phải gửi tiết kiệm bao nhiêu tháng? (t, k, max đều nhập từ bàn phím).

9. Viết chương trình tìm các số a, b, c thỏa mãn: $\overline{abc} = \overline{ab} + \overline{bc} + \overline{ca}$

10. Viết chương trình phân tích một số nguyên thành tích của các số nguyên tố.

11. Viết chương trình giải bài toán:

" Vừa gà vừa chó,
Bó lại cho tròn
Ba mươi sáu con.
Một trăm chân chẵn".

Hỏi có bao nhiêu con gà, bao nhiêu con chó?

12. Viết câu lệnh rẽ nhánh tính:

a. $z = \begin{cases} x^2 + y^2 & nếu\ x^2 + y^2 \le 1. \\ x = y & nếu\ x^2 + y^2 > 1\ và\ y \ge x. \\ 0{,}5 & nếu\ x^2 + y^2 > 1\ và\ y < x. \end{cases}$

b.

$$z = \begin{cases} |x| + |y| \ nếu\ điểm\ (x,y) thuộc\ hình\ tròn\ có\ bán\ kính\ r(r > 0), tâm\ (a,b). \\ x + y\ trong\ trường\ hợp\ còn\ lại. \end{cases}$$

13. Lập trình tính:

a. $Y = \sum_{n=1}^{50} \frac{n}{n+1}$

b. $e(n) = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!} + \cdots, cho\ đến\ khi\ \frac{1}{n!} < 2 \times 10^{-6}$

Đưa giá trị e(n) ra màn hình.

14. Nhập từ bàn phím tuổi của cha và con (hiện tại tuổi cha lớn hơn hai lần tuổi con và tuổi cha hơn tuổi con ít nhất là 25). Đưa ra màn hình câu trả lời cho câu hỏi "Bao nhiêu năm nữa thì tuổi cha gấp đôi tuổi con?"

15. Một người gửi tiết kiệm không kì hạn với số tiền A đồng với lãi suất 0,3% mỗi tháng. Hỏi sau bao nhiêu tháng, người đó rút hết số tiền thì sẽ nhận được số tiền ít nhất là B đồng? Biết rằng với việc gửi tiết kiệm không kì hạn thì lãi không được cộng vào vốn.

HƯỚNG DẪN GIẢI BÀI TẬP

1. - Nhập vào 3 số a,b,c
   - Khởi gán max:=a;
   - So sánh max với b,c nếu max bé hơn thì gán lại max.
2. - Nhập các hệ số a,b,c
   - Tính dt:=b*b - 4*a*c;
   - Sử dụng lệnh If biện luận nghiệm theo dt và thông báo kết quả.
3. - Nhập các hệ số a,b,c
   - Sử dụng lệnh If biện luận: Nếu đồng thời xảy ra a+b>c, b+c>a, c+a>b thì thông báo a,b,c đúng là độ dài 3 cạnh của một tam giác nào đó. Ngược lại thì không phải.
4. - Nhập vào số KW điện tiêu thụ.
   - Dùng lệnh If hoặc case để biện luận và tính tiền theo gía thành.
   - Thông báo kết quả.
5. - Đoạn chương trình trên hoàn toàn đúng cú pháp;
   - Ý nghĩa của nó là cho phép nhập vào hai số nguyên a,b. Nếu a <b thì sẽ thông báo lên màn hình bất dẳng thức đó, ngược lại thông báo bất đẳng thức b <= a.
8. - Nhập vào một số nguyên dương.
   - Dùng lệnh Repeat hoặc While để lặp lại công việc sau đây cho đến khi n=0 thì dừng:
     + Ghi lên màn hình chữ số cuối cùng của n (n mod 10) bằng lệnh write;
     + Loại bỏ chữ số cuối cùng bằng cách gán lại n:= n div 10.
9. - Nhập số m, n nguyên dương (m<n);

- Dùng lệnh for lần lượt duyệt tất cả các số từ m đến n, nếu đúng là số hoàn hảo thì thông báo, nếu không thì bỏ qua.

10. - Nhập số n.

- Khởi gán i:=2;

- Sử dụng lệnh While lặp lại: Chừng nào i<=round(sqrt(n)) và n mod i <>0 thì tăng i lên 1; Sau đó kiểm tra nếu đồng thời i<>1 và i > round(sqrt(n))i n thì n là số nguyên tố; Ngược lại n không phải là số nguyên tố.

11. - Nhập số n nguyên dương;

- Dùng lệnh for lần lượt duyệt tất cả các số từ 1 đến n, nếu đúng là số nguyên tố thì thông báo nếu không thì bỏ qua.

12. - Khởi gán a:=1, b:=1;

- Dùng lệnh Repeat lặp lại:

c:=a+b;

a:=b; b:=c;

Cho đến khi c> 2000 thì dừng lại.

- Thông báo kết quả là a.

13. - Nhập t, k, max.

- Khởi gán st:=0; (st là số tháng).

- Dùng lệnh Repeat lặp lai:

+ st:=st+1;

+ t:=t+t*k;

. Cho đến khi t >= max thì dừng.

- Thông báo kết quả là st.

14. - Gọi số cần tìm có các chữ số là a,b,c.

- Sử dụng 3 lệnh for lồng nhau để thử hết tất cả các khả năng của a,b,c từ 1 đến 9, nếu thỏa mãn 100*a+10*b+c=11*(a+b+c) thì thông báo kết quả.

15. - Nhập n.

- Khởi gán i:=2.

- Ghi ra màn hình write('n= ');

- Dùng lệnh While lặp lại:

```
While i<= n do
  If n mod i = 0 then
          begin
                If  i<n then write(i,'*')
                        Else write(i);
```

```
            n:= n div i;
        end
    else i:= i+1;
```

16. - Khai báo biến g để chứa số gà.

- Sử dụng lệnh for duyệt hết tất cả các khả năng của g (từ 1 đến 36)

Nếu 2*g+4*(36-g)=100(*) thì thông báo Số gà là g, số chó là 36 - g.

Chú ý: Có thể khai báo thêm biến dem kiểu nguyên và khởi gán dem:=0 để đếm số nghiệm, cứ mỗi lần đẳng thức (*) thõa mãn thì tăng biến đếm lên 1. Cuối cùng sẽ thông báo bài toán có dem nghiệm; trường hợp dem = 0 tức là vô nghiệm.

CHƯƠNG 5. KIỂU DỮ LIỆU TẬP HỢP, KIỂU MẢNG, KIỂU XÂU

Mở đầu:

Nội dung chương nầy đưa ra các kiểu dữ liệu có cấu trúc (Set, Array, String). Trong mỗi kiểu dữ liệu ra cú pháp của từng kiểu, cách định nghĩa và khai báo chúng. Đồng thời mô tả các thao tác trên từng kiểu bằng các ví dụ minh họa.

Mục tiêu: Học xong chương này Sinh viên

- Hiểu được dữ liệu kiểu tập hợp, kiểu mảng, kiểu xâu.
- Biết cách khai báo cho từng kiểu dữ liệu.
- Hiểu được cấu trúc của từng kiểu dữ liệu
- Biết cách sử dụng kiểu dữ liệu cho phù hợp với yêu cầu bài toán.

5.1. Kiểu tập hợp

5.1.1. Khái niệm:

Tập hợp là kiểu dữ liệu có cấu trúc bao gồm các phần tử có cùng kiểu vô hướng đếm được nào đó.

5.1.2. Định nghĩa kiểu vàkhai báo biến:

Định nghĩa kiểu:

```
Type  ten_kieu = Set  of  tap_hop;
```

Giải thích: Type là từ khóa để khai báo kiểu;

+ Ten_kieu do người sử dụng tự đặt lấy đúng theo quy định của Turbo Pascal;

+ Set of là từ khóa để khai báo kiểu tập hợp.

+ Tap_hop Là một tập hợp gồm các phần tử có cùng kiểu vô hướng đếm được, ví dụ: 1..10; byte; 'a'.. 'z'; ...

Khai báo biến: Sau khi đã khai báo kiểu ta có thể khai báo biến thông qua tên kiểu.

Ví dụ:

```
Type  chu_cai = set  of  'a'..'z';
      Tuoi = set  of  0..250;
Var   ch: chu_cai ;
      t1,t2 : tuoi;
```

Cũng có thể khai báo biến tập hợp trực tiếp không thông qua khai báo kiểu như sau:

```
Var  ten_bien :  Set  of  tap_hop;
```

Ví dụ:

```
Var     ch : set of 'a'..'z';
        t1,t2 : set of 1..250;
```

Chú ý:

(1). Tap_hop cho phép tối đa là 256 phần tử.

(2). Giá trị của một biến kiểu tập hợp là một tập hợp con của tap_hop, khi biểu diễn các phần tử đó phải được đặt trong cặp dấu móc vuông " [..]".

(3). Tap_hop có thể biểu diễn dưới dạng kiểu đoạn con, kiểu liệt kê, các kiểu cơ sở như byte; char...

(4). Tập hợp rỗng được biểu diễn là "[]".

5.1.3. Các phép toán:

a. Các phép quan hệ:

Cho a,b là các biến có cùng một kiểu tập hợp khi đó ta có các phép toán:

a=b Khi a và b bằng nhau (theo nghĩa tập hợp).

a<>b Khi tập hợp a khác tập hợp b.

a<=b Khi a là tập con của b.

a>=b Khi b là tập con của a.

Chú ý: Không tồn tại phép ">" hay mhỏ hơn "<". Các phép này có thể được biểu diễn thông qua các phép quan hệ đã trình bày ở trên. Ví dụ để có được so sánh a< b ta có thể sử dụng cách viết: (a <= b) and (a<>b)...

b. Phép toán IN:

Phép toán IN dùng để kiểm tra xem một phần tử nào đó có mặt trong một tập hợp hay không?

Ví dụ: 'a' in ['a'..'z'] cho giá trị là true.

c. Các phép toán tập hợp:

Phép hợp được biểu diễn bởi dấu "+", ví dụ A ∪ B được viết là A + B.

Phép giao được biểu diễn bởi dấu "*", ví dụ A ∩ B được viết là A * B.

Phép hiệu được biểu diễn bởi dấu " -", ví dụ A \ B được viết là A - B.

5.1.4. Ví dụ:

Viết chương trình nhập một chữ cái rồi cho biết đó là nguyên âm hay là phụ âm.

Hướng dẫn:

- Khai báo biến chu_cai và biến nguyen_am kiểu tập hợp mà các phần tử thuộc kiểu Char; khai báo biến ch kiểu char để chứa chữ cái nhập vào.

- Định nghĩa tập chữ cái gồm các phần tử từ 'a' đến 'z' và 'A' đến 'Z'; nguyên âm gồm các chữ 'A','E','I','O','U'.

- Nhập một chữ cái. Sử dụng phép toán IN để kiểm tra và thông báo kết quả

Chương trình:

```
Program  NguyenAm;
```

```
Var chu_cai, nguyen_am : set  of  char;
Ch:char;
Begin
     chu_cai :=['a'..'z','A'..'Z'];
     Nguyen_am :=['A','E','I','O','U'];
     Repeat
               Write('nhap mot chu cai: ');
               Readln(ch);
     Until  ch  in  chu_cai;
     If  upcase(ch)  in  nguyen_am then writeln(ch,' la
     nguyen am')
     Else writeln(ch, ' la phu am');
     Readln
End.
```

5.2. Kiểu mảng

5.2.1. Khái niệm:

Mảng là kiểu dữ liệu có cấu trúc bao gồm một số cố định các phần tử có cùng kiểu, cùng một tên chung. Mỗi phần tử của mảng có thể xem như là một biến.

5.2.2. Định nghĩa kiểu và khai báo biến:

- *Định nghĩa kiểu*:

```
Type  ten_kieu = Array[CTCS] of  KPT;
```

Giải thích:

+ Type là từ khóa để khai báo kiểu;

+ Ten_kieu do người sử dụng tự đặt lấy đúng theo quy định của Turbo Pascal;

+ CTCS ký hiệu cho các tập chỉ số.

+ KPT ký hiệu cho kiểu phần tử và phải là kiểu đơn giản chuẩn, cũng có thể là kiểu mảng...

- Array, of là từ khóa để khai báo kiểu mảng.

- *Khai báo biến:* Sau khi đã khai báo kiểu ta có thể khai báo biến thông qua tên kiểu.

Ví dụ:

```
Type  Mang_nguyen = array[1..100] of integer;
      Mang_thuc  = array[1..50]  of  real;
Var   m: mang_nguyen ;
      a,b : mang_thuc;
```

Khi đó m là một biến kiểu mảng nên có các phần tử: m[1], m[2],...,m[100]. Mỗi phần tử như thế có thể xem như là một biến kiểu số nguyên và được biểu diễn thông qua tên biến và chỉ số (đặt trong cặp dấu móc vuông "[...]").

Tương tự a, b cũng có các phần tử là a[1], a[2],...,a[50]; b[1], b[2],...,b[50].

Cũng có thể khai báo biến mảng trực tiếp không thông qua khai báo kiểu như sau:

```
Var  ten_bien: Array[CTCS] of  KPT;
```

Ví dụ:

```
Var  m : array[1..100]  of  integer;
     a,b : array[1..50]  of  real;
```

Chú ý:

(1). Các tập chỉ số phải là các kiểu dữ liệu vô hướng đếm được và chỉ gồm hữu hạn phần tử. Ví dụ kiểu đoạn con, kiểu liệt kê, kiểu byte ...

(2). Nếu có nhiều tập chỉ số thì dùng dấu phẩy "," để ngăn cách. Có bao nhiêu tập chỉ số thì mảng tương ứng được gọi là bấy nhiêu chiều và số phần tử của mảng bằng tích số phần tử của các tập chỉ số đó.

Ví dụ:

```
Var     m :  array[1..10,1..5]  of  integer;
```

Khi đó m có 50 phần tử (10*5), các phần tử của biến mảng m là:

m[1,1], m[1,2],..,m[1,5].

m[2,1], m[2,2],..,m[2,5].

m[10,1], m[10,2],..,m[10,5].

(4). Không thể đọc và ghi dữ liệu cho toàn biến kiểu mảng mà phải thực hiện thông qua từng phần tử của biến mảng.

5.2.3. Các thao tác đối với biến kiểu mảng:

a. Gán một biến mảng cho một biến mảng khác:

Cho a, b là các biến có cùng một kiểu mảng và giả sử các phần tử của biến a đã được gán giá trị, khi đó ta có thể dùng lệnh gán b:=a; Sau lệnh gán này các phần tử của biến mảng b có giá trị tương ứng giống như các phần tử của biến mảng a.

Ví dụ:

```
Var  a,b : array[1..5] of byte;
          i:byte;
Begin
     For i:=1 to 5 do  a[i] := i;
     b:=a;
...
End.
```

Thực hiện chương trình này ta có:

a[1]=b[1]=1;

a[2]=b[2]=2;

```
...
a[5]=b[5]=5;
```

b. Sắp xếp các phần tử của mảng:

Giả sử ta có một biến mảng chứa các phần tử nào đó có kiểu vô hướng đếm được, vấn đề đặt ra là hãy sắp xếp các phần tử đó theo thứ tự tăng dần hay giảm dần.

Đây là bài toán sắp xếp cơ bản, người ta đã tìm ra nhiều thuật toán sắp xếp hay như thuật toán sắp xếp "nổi bọt", thuật toán "chèn",... Tuy nhiên ở đây ta đưa ra một cách sắp xếp đơn giản sau:

Giả sử cần sắp xếp các phần tử a[1], a[2],...,a[n] theo thứ tự tăng dần ta dùng đoạn chương trình sau:

```
...
 For i:=1 to n-1 do
      For j:=i+1 to n do
           if  a[i] > a[j] then
                begin
                     tg:=a[i];
                     a[i]:=a[j];
                     a[j]:=tg;
                end;
...
```

Trong đoạn chương trình trên, với mỗi giá trị của i ta sử dụng lệnh for với biến điều khiển j để lần lượt so sánh a[i] với tất cả các phần tử a[j] đứng sau nó, nếu a[i] lớn hơn a[j] thì ta đổi chỗ a[i] và a[j] cho nhau, như vậy thì phần tử a[i] sẽ đứng đúng vị trí của nó trong danh sách sắp xếp. Cho i lần lượt đi từ 1 đến n-1 ta sẽ có danh sách được sắp xếp hoàn chỉnh.

Hoàn toàn tương tự cho vấn đề sắp xếp giảm dần.

5.2.4. Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào n số nguyên dương rồi tính trung bình cộng và trung bình nhân của nó.

Hướng dẫn:

- Khai báo biến n, i kiểu số nguyên. Biến mảng m kiểu số nguyên để chứa dãy số nhập vào. Các biến tbc,tbn kiểu số thực để chứa kết quả.

- Sử dụng lệnh for tính tổng, tích của n số nguyên dương đó từ đó tính trung bình cộng và trung bình nhân.

- Thông báo kết quả lên màn hình.

Chương trình:

```
 Program  Tinh_tb;
```

```
Var n,i: integer;
     tbc,tbn :real;
     a:array[1..100] of  integer;
Begin
     Repeat
          Write(' nhap n : ');
          Readln(n);
     Until  n>0;
     tbc:=0; tbn:=1;
     For  i:=1 to n do
          Begin
             Repeat
                  Write('a[',i,']= ');
                  Readln(a[i]);
             Until a[i] >0;
             Tbc:=tbc+a[i];
             Tbn:=tbn*a[i];
          End;
     tbc:= tbc/n; tbn:= exp(1/n*ln(tbn));
     writeln('trung binh cong cua day la:', tbc:8:2);
     writeln('trung binh nhan cua day la:', tbn:8:2);
     Readln
End.
```

Ví dụ 2: Viết chương trình nhập vào một dãy 10 số nguyên khác nhau tùng đôi một. Hãy chỉ ra độ dài của dãy con gồm các phần tử liên tiếp tăng mà có số phần tử nhiều nhất.

Ví dụ: Nếu nhập vào dãy: 2, 4, 1, 5, 8, 7, 9. Thì độ dài cần tìm là 3 (của dãy con:

1, 5, 8).

Hướng dẫn:

- Khai báo biến i,j, kiểu số nguyên. Biến mảng m kiểu nguyên để chứa dãy số nguyên nhập vào. Biến d để chứa độ dài của dãy con tạm thời, Biến max để chứa độ dài cực đại của dãy con đến thời điểm đang xét.

-Nhập dãy số: Mỗi lần nhập a[i] ta dùng lệnh for j kiểm tra xem chúng có trùng vào các số từ a[1] đến a[i-1] hay không, nếu trùng thì yêu cầu nhập lại.

- Khởi gán d:=1; max:=1;

- Dùng lệnh for duyệt i từ 2 đến n:

Nếu a[i-1] < a[i] thì tăng d lên 1.

Nếu a[i-1] > a[i] thì:

Kiểm tra xem nếu d > max thì gán lại max := d;

Sau đó gán lại d:=1;

- Tuy nhiên có thể dãy con lớn nhất nằm cuối cùng nên ta lại phải kiểm tra nếu max< d thì gán lại max :=d;

- Thông báo kết quả max lên màn hình.

Chương trình:

```
Program  Tim_dd_cucdai;
Var i,j,d,max: integer; ok : Boolean;
     a:array[1..10] of  integer;
Begin
     For  i:=1 to 10 do
          repeat
             Write('a[',i,']= ');
             Readln(a[i]);
             ok:=true;
             for j:=1 to i-1 do
                   if a[j]=a[i] then
                         begin
                              ok:=false;
                              break;
                         end;
          until ok;
     d:= 1; max:=1;
     For  i:=2 to 10 do
          Begin
               if  a[i-1] > a[i] then
                    begin
                         max:=t;
                         d:=1;
                    end;
               else d:=d+1;
          end;
     If max < d then max:=d;
     writeln('Day  con  tang  co  do  dai  lon  nhat  la:',
max);
     Readln
End.
```

Ví dụ 3: Viết chương trình nhập vào hai ma trận vuông cấp n (với các phần tử nguyên) rồi tính tổng và tích của hai ma trận đó.

Hướng dẫn:

- Khai báo biến n, i, j, k kiểu số nguyên. Biến mảng a, b, c, d hai chiều các phần tử kiểu số nguyên để chứa các ma trận nhập vào và kết quả.

- Nhập cấp n của các ma trận vuông.

- Dùng hai lệnh for lồng nhau để nhập các phần tử cho ma trận a và b.
- Tính các phần tử của ma trận tổng c và ma trận tích d theo công thức.
- Thông báo kết quả lên màn hình.

Chương trình:

```
Program  ma_tran;
Var n,i,j,k: integer;
     a,b,c,d :array[1..20,1..20] of  integer;
Begin
     Repeat
          Write(' nhap n : ');Readln(n);
     Until (n>0)and(n <= 20);
     Writeln('nhap ma tran a:');
     For  i:=1 to n do
          For  j:=1 to n do
               Begin
                    Write('a[',i,',',j,']= ');
                    Readln(a[i,j]);
               End;
     Writeln('nhap ma tran b: ');
     For  i:=1 to n do
          For  j:=1 to n do
               Begin
                    Write('b[',i,',',j,']= ');
                    Readln(b[i,j]);
               End;
     For  i:=1 to n do
          For  j:=1 to n do
               Begin
                 c[i,j]:= a[i,j] + b[i,j];
                 d[i,j]:= 0;
                 For k:=1 to n do
                    d[i,j]:= d[i,j] + a[i,k]* b[k,j];
               End;
     Writeln('ma tran tong la:');
     For  i:=1 to n do
          Begin
               For  j:=1 to n do Write(c[i,j]:5);
               writeln;
          End;
     Writeln('ma tran tich la:');
     For  i:=1 to n do
          Begin
               For  j:=1 to n do Write(d[i,j]:5);
```

```
                writeln;
            End;
        Readln
End.
```

5.3. Kiểu xâu

5.3.1. Khái niệm:

Xâu là kiểu dữ liệu chuẩn bao gồm một dãy các ký tự trong bảng mã ASCII có độ dài tối đa là 255 ký tự.

5.3.2. Định nghĩa kiểu và khai báo biến:

- *Định nghĩa Kiểu*:

```
Type  ten_kieu = String[n];
```

Giải thích:

+ Ten_kieu do người sử dụng tự đặt đúng theo quy định của Turbo Pascal.

+ String là từ khóa để khai báo kiểu xâu.

+ n là độ dài tối đa của xâu.

- *Khai báo biến:* Sau khi đã khai báo kiểu ta có thể khai báo biến thông qua tên kiểu.

```
Var  ten_bien : ten_kieu;
```

Ví dụ:

```
Type  St  = string[50];
Var  st1,st2 : st;
     st3  : string;
```

Cũng có thể khai báo biến kiểu xâu trực tiếp không thông qua định nghĩa kiểu như sau:

```
Var  ten_bien : string[n];
```

Ví dụ:

```
Var     st1,st2 : string[50];
```

Chú ý:

(1). Một biến kiểu string[n] sẽ được cấp phát n+1 byte bộ nhớ, trong đó byte đầu tiên lưu độ dài thực sự của xâu (số đó chính là: ord(st[0]).

Ví dụ: st:='abc123' sẽ được lưu như sau:

st[0]	a	b	c	1	2	3

(2). Có thể sử dụng lệnh read, readln để nhập dữ liệu, lệnh write, writeln để ghi dữ liệu kiểu xâu, cũng có thể sử dụng lệnh gán như các biến khác.

(3). Việc truy xuất từng ký tự của xâu phải thông qua tên biến cùng với chỉ số của nó (chỉ số truy xuất phải đặt trong cặp dấu móc vuông "[...]"). Chẳng hạn với ví dụ ở mục (1) thì ta có: st[1]='a'; st[2]='b';... st[6] = '3';

(4). Xâu không có ký tự nào được gọi là xâu rỗng và được biểu diễn là: ''.

5.3.3. Các phép toán trên xâu:

a. Phép nối xâu ("+"):

Giả sử st1, st2 là các xâu khi đó st1 + st2 cho ta một xâu bằng cách nối theo thứ tự xâu st1 với xâu st2.

Ví dụ: st1 := 'abcd'; st2 := 'ABC', khi đó st1 + st2 cho kết quả là xâu 'abcdABC'.

b. Phép so sánh xâu:

Hai xâu có độ dài bằng nhau: Lần lượt từng cặp ký tự tương ứng của hai xâu sẽ được so sánh với nhau từ trái sang phải; Nếu gặp cặp ký tự khác nhau đầu tiên thì xâu nào có ký tự lớn hơn là lớn hơn. Nếu mọi cặp ký tự đều giống nhau thì hai xâu bằng nhau.

Ví dụ: 'abce' > 'abcd'.

Hai xâu có độ dài khác nhau: Lần lượt từng cặp ký tự tương ứng của hai xâu sẽ được so sánh với nhau từ trái sang phải cho đến ký tự cuối cùng của xâu ngắn; Nếu gặp cặp ký tự khác nhau đầu tiên thì xâu nào có ký tự lớn hơn là lớn hơn. Nếu không thì xâu có độ dài lớn hơn là lớn hơn.

Ví dụ: 'abcde' > 'abcd'.

'abacde' < 'abcd'

5.3.4. Các thủ tục và hàm trên xâu ký tự:

a. Các thủ tục:

Ký hiệu:

St, st1,st2: là biến kiểu xâu.

Num: là biến kiểu số nguyên chỉ số ký tự.

Pos: là biến kiểu số nguyên chỉ vị trí nào đó trong xâu.

- `Delete(st, pos, num)`

Thủ tục này sẽ xóa trong xâu st đi num ký tự bắt đầu từ vị trí pos.

Ví dụ: st := 'abcde';

Delete(st, 2, 3) ----→khi đó giá trị của st là: 'ae'.

- `Insert(st2,st1,pos)`

Thủ tục này sẽ chèn xâu st2 vào xâu st1 bắt đầu từ vị trí pos.

Ví dụ: st1:='abc'; st2:='deft';

Insert(st2,st1, 3) ----→ khi đó st1 có giá trị là: 'abdeftc'.

- Str(value, st)

Trong đó value là một biểu thức số học có thể ghi dưới dạng có quy cách.

Thủ tục này sẽ đổi giá trị của số (value) thành xâu và gán cho biến xâu st.

Ví dụ: str(123.456,st) ----→ khi đó st sẽ có giá trị là: '123.456'.

- Val(st,var, code)

Trong đó st là một xâu biểu diễn số.

+ Var là một biến kiểu số nguyên hay số thực.

+ Code là biến kiểu số nguyên.Thủ tục này sẽ đổi xâu st biểu diễn số thành số và gán cho biến var, nếu việc đổi thành công thì biến code có giá trị bằng 0, ngược lại nếu xâu st biểu diễn số bị lỗi thì khi đó var không xác định còn code có giá trị là vị trí lỗi trong biểu diễn số của xâu st.

Ví dụ: st1:= '123.456';

str(st1, a, k) ----→ khi đó a sẽ có giá trị là 123.456.

st2:= '12q.456';

str(st2, a, k) ----→ khi đó a không có giá trị xác định còn code có giá trị là 3 chính là vị trí của q.

b. Các hàm trên xâu ký tự:

- Length(st)

Cho kết quả là độ dài thực sự của xâu st.

Ví dụ: st:='abcd';

length(st) cho kết quả là 4.

Chú ý: length(st) = ord(st[0]);

- Copy(st,pos,num)

Cho kết quả là xâu con của xâu st gồm num ký tự bắt đầu từ vị trí pos.

Ví dụ: copy('abcdef',3,2) cho kết quả là xâu 'cd'.

- Concat(st1,st2,...,stn) (n>=2)

Cho kết quả là xâu được ghép nối theo thứ tự các xâu st1, st2, ..., stn.

Ví dụ: Concat('abc','de','123') cho kết quả là xâu 'abcde123'.

Chú ý: Nếu xâu kết quả có độ dài > 255 thì máy sẽ báo lỗi.

- Pos(st1,st2)

Cho kết quả là vị trí xuất hiện đầu tiên của xâu st1 trong xâu st2. Nếu xâu st1 không có mặt trong xâu st2 thì hàm cho kết quả bằng 0.

Ví dụ: Pos('ab','deabcab') cho kết quả là 3.

Pos('ab','defbc') cho kết quả là 0.

5.3.5. Các ví dụ:

Ví dụ 1: Viết chương trình nhập vào một xâu ký tự bất kỳ và in ra màn hình xâu ngược lại.

Hướng dẫn:

- Khai báo biến st kiểu xâu để nhập dữ liệu, biến i điều khiển lệnh for.

- Nhập xâu st.

- Dùng lệnh for dạng lùi để ghi các ký tự của xâu từ sau tới.

Chương trình:

```
Program  xau_nguoc;
Var  st: string;
           i:byte;
Begin
      Write('nhap xau bat ky:'); readln(st);
      Writeln('xau nguoc cua xau ', st, 'la: ');
      For i:=length(st) downto  1 do write(st[i]);
      Readln
End.
```

Ví dụ 2: Một xâu được gọi là Palindrom nếu nó trùng với xâu đảo. Viết chương trình nhập vào một xâu bất kỳ và cho biết nó có phải là Palindrom hay không?

Hướng dẫn:

- Khai báo biến st, st1 kiểu xâu để nhập dữ liệu và chứa xâu đảo, biến i điều khiển lệnh for.

- Nhập xâu st.

- Khởi gán st1 := '';

- Dùng lệnh for dạng lùi để lần lượt bổ sung các ký tự của xâu st từ sau tới cho xâu st1.

- So sánh hai xâu và thông báo kết quả.

Chương trình:

```
Program  Palindrom;
Var  st,st1: string;
           i:byte;
Begin
     Write('nhap xau bat ky:'); readln(st); St1:='';
     For i:=length(st) downto  1 do st1:= st1 + st[i];
     If st=st1 then Writeln(st, ' la xau Palindrom  ')
     Else Writeln(st,' khong phai  la Palindrom  ');
     Readln
End.
```

BÀI TẬP CHƯƠNG 5

5.1. Kiểu mảng:

1. Viết chương trình nhập vào một dãy số nguyên và thực hiện các yêu cầu sau:
   a. In ra màn hình các số chẵn và tổng lập phương của chúng.
   b. Giá trị lớn nhất và nhỏ nhất của dãy đó.
   c. Số âm lớn nhất và số dương nhỏ nhất của dãy.
   d. Số âm đầu tiên và chỉ số của nó.
   e. Số dương cuối cùng và chỉ số của nó.

2. Viết chương trình nhập một dãy n số nguyên và thực hiện dịch chuyển k phần tử đầu dãy về cuối dãy (k<n).

Ví dụ n=5, k=3 ta nhập vào dãy 3, 5, 7,1, 2. Dãy cần tìm sẽ là 1, 2, 3, 5, 7.

3. Viết chương trình tìm phần tử thứ n của dãy số FIBONACCI.

4. Viết chương trình nhập vào bậc n và các hệ số của một hàm đa thức. Hãy cho biết giá trị của hàm đó tại một điểm bất kỳ.

5. Viết chương trình nhập một dãy số đôi một khác nhau và:
   a. Cho biết độ dài lớn nhất của dãy gồm các số 0 liên tiếp.
   b. Tìm dãy con liên tiếp tăng và có tổng lớn nhất.

6. Viết chương trình:
   a. Bổ sung một phần tử vào mảng 1 chiều.
   b. Loại bỏ một phần tử ra khỏi mảng 1 chiều.

7. Viết chương trình nhập vào một dãy số và cho biết dãy đó có bao nhiêu phần tử đôi một khác nhau đồng thời thông báo các phần tử đod lên màn hình.

8. Viết chương trình tìm cách lấy ra n nghìn đồng từ các loại tiền 1 nghìn, 2 nghìn, 5 nghìn, 10 nghìn, 20 nghìn, 50 nghìn, 100 nghìn sao cho tổng số tờ là bé nhất. (Số tiền n, số tờ của mỗi loại tiền được nhập từ bàn phím).

9. Viết chương trình nhập vào một ma trận vuông cấp n rồi cho biết:
   a. Tổng tất cả các phần tử nằm trên đường chéo chính.
   b. Tổng tất cả các phần tử nằm phía trên đường chéo chính.

10. Viết chương trình nhập vào một ma trận cấp mxn rồi tính tổng tất cả các phần tử mà thỏa mãn i - j = k. (m,n,i,j,k nhập từ bàn phím, 0<i,j,k<n).

11. Viết chương trình xổ số với cơ cấu như sau:
   - Ba giải tư, quay 2 số.
   - Hai giải ba, quay 3 số.
   - Một giải nhì, quay 4 số.
   - Một giải nhất, quay 5 số.

- Một giải đặc biệt, quay 6 số.

5.2. Kiểu xâu:

12. Viết chương trình nhập một xâu bất kỳ, in ra màn hình xâu sau khi đã xóa bỏ hết các ký tự trắng dư thừa.

(Ký tự trắng dư thừa là ký tự trắng đứng ở đầu xâu, cuối xâu hoặc đi liền nhau).

13. Viết chương trình nhập vào một xâu bất kỳ và:

a. In ra màn hình xâu sau khi đã đổi tất cả chữ thường của xâu thành chữ hoa.

b. Cho biết xâu đó có bao nhiêu từ. (Từ là một dãy các ký tự khác ký tự trắng liên tiếp. Ví dụ: Xâu ' cong hoa xa hoi ' có 4 từ).

14. Viết chương trình nhập một xâu là họ và tên của một người nào đó, in ra màn hình tên của người đó.

15. Viết chương trình đổi số nguyên n thành số nhị phân. (Số nhị phân được biểu diễn dưới dạng xâu).

16. Viết chương trình nhập vào một xâu rồi hiển thị mỗi từ trên một hàng (theo thứ tự xuất hiện) đồng thời cho biết tổng số từ có trong xâu trên.

17. Viết chương trình nhập vào 2 xâu biểu diễn số nguyên. Thực hiện tính tổng của của 2 xâu đó như là việc tính tổng của 2 số nguyên.

18. Viết chương trình nhập từ bàn phím số nguyên dương N (N≤100) và dãy A gồm N số nguyên A1, A2, ..., An có giá trị tuyệt đối không lớn hơn 1000. Hãy cho biết dãy A có phải là cấp số cộng hay không và thông báo kết quả ra ngoài màn hình.

19. Viết chương trình nhập từ bàn phím số nguyên dương N (N≤100) và dãy A gồm N số nguyên A1, A2, ..., An có giá trị tuyệt đối không lớn hơn 1000. Hãy đưa ra những thông tin sau:

a. Số lượng số chẵn và số lẻ trong dãy.

b. Số lượng số nguyên tố trong dãy.

20. Chương trình sau đây thực hiện những gì?

```
program bt8;
const NMax = 50;
type Mass = array[1..NMax,0..NMax-1] of real;
var a: Mass;
    i, j, N: byte; C:real;
begin
    write('nhap N = '); readln(N);
    for i:= 1 to N do
        for j:= 0 to N-1 do
            begin
                write('A[',i,',',j,']='); readln(a[i,j]);
            end;
```

```
    for i:= 1 to N do
        for j:= 0 to N-1 do
            begin
            C:= a[i,j];
            a[i,j]:= a[N-i+1],j];
            a[N-i+1],j]:=C;
            end;
    for i:=1 to n do
        begin
            for j:=0 to N-1 do write(a[i,j]:5:2);
            writeln;
        end;
    readln
end.
```

21. Cho mảng hai chiều có kích thước nxn với các phần tử là những số nguyên. Tìm trong mỗi hàng phần tử lớn nhất rồi đổi chỗ nó với phần tử có chỉ số hàng bằng chỉ số cột.

Chương trình sau đây giải bài toán trên.

```
program diag;
var
    n, i, j, max, ind, vsp: integer;
    a: array[1..15,1..15] of integer;
begin
    write('nhap N nho hon 15: '); readln(n);
    for i:=1 to n do
        for j:=1 to n do
            begin
                write('A[',i,',',j,']= ');
                readln(a[i,j]);
            end;
    for i:=1 to n do
        begin
            max:= a[i,1]; ind:=1;
            for j:=2 to n do
                if a[i,j] > max then
                    begin
                        vsp:= a[i,i];
                        a[i,i]:= max;
                        a[i,ind]:= vsp;
                    end;
        end;
    for i:=1 to n do
```

```
        begin
            writeln;
            for j:=1 to n do write(a[i,j]:3);
        end;
    readln;
end.
```

Hãy sửa lại chương trình trên khi thay yêu cầu tìm kiếm trong mỗi hàng bằng tìm kiếm trong mỗi cột.

22. Hãy viết chương trình nhập từ bàn phím xâu kí tự S có độ dài không quá 100. Hãy cho biết có bao nhiêu chữ số xuất hiện trong xâu S. Thông báo kết quả ra màn hình.

HƯỚNG DẪN GIẢI BÀI TẬP

1. - Khai báo biến mảng để nhập dãy số.

a. - Khai báo biến s và khởi gán s:=0 để chứa tổng lập phương của dãy. Sử dụng lệnh for lần lượt duyệt qua tất cả các phần tử nếu là số chẵn thì ghi ra màn hình đồng thời cộng dồn vào cho biến s.

b. - Gán max (min) bằng phần tử đầu tiên rồi lần lượt so sánh với tất cả các phần tử còn lại nếu max (min) bé hơn (lớn hơn) thì gán lại.

c. - Dùng lệnh while để duyệt lần lượt các phần tử cho đến khi gặp phần tử âm đầu tiên thì dừng lại và gán giá trị của nó cho ammax, sau đó lại tiếp tục duyệt hết các phần tử còn lại nếu gặp số âm mà lớn hơn ammax thì gán lại. Nếu duyệt hết tất cả các phần tử của dãy mà không có số âm nào thì thông báo "khong co so am".

Hoàn toàn tương tự cho việc tìm số dương bé nhất.

d. - Dùng lệnh while để duyệt lần lượt các phần tử cho đến khi gặp phần tử âm thì dừng lại và thông báo giá trị và chỉ số của số âm đó. Nếu duyệt hết tất cả các phần tử của dãy mà không có số âm nào thì thông báo "khong co so am".

e. - Tương tự câu d. (lưu ý là duyệt từ sau duyệt tới).

2. - Khai báo mảng a để nhập dữ liệu, mảng b làm trung gian.

- Nhập dãy số (lưu vào a).

- Sử dụng lệnh for gán giá trị của các phần tử của mảng a cho các phần tử của

for i:=k+1 to n do b[i-k]:=a[i];

for i:=1 to k do b[i+k]:=a[i];

- Gán lại a:=b;

3. - Khai báo một biến mảng để chứa các phần tử của dãy Fibonacci.

- Khởi gán a[1]:=1; a[2]:=1;

- Dùng lệnh for để tính a[i]:= a[i-1] + a[i-2] (i=3,4,...,n).

- Thông báo kết quả a[n].

4. - Nhập n và các hệ số a[1],a[2],...,a[n].

- Nhập giá trị cho biến x; khởi gán giá trị hàm y:=a[n].

- Sử dụng lệnh lặp for :

for i:=n downto 1 do y:=y*x+a[i-1];

- Thông báo kết quả y.

5. - Nhập dãy số.

- Khai báo một biến t để chứa độ dài tạm thời của các dãy con các phần tử 0 liên tiếp; biến max để chứa độ dài cực đại và khởi gán chúng bằng 0.

- Dùng lệnh for để duyệt lần lượt các phần tử của dãy.

+ Nếu a[i] <> 0 thì

So sánh nếu max < t thì gán lại max:=t và t:=0;

+ Ngược lại a[i] = 0 thì t:= t + 1;

- Cuối cùng so sánh nếu max < t thì gán lại max:=t;

- Thông báo kết quả max.

6. a.- Nhập mảng n phần tử ; nhập vị trí cần chèn k và giá trị cần chèn là x.

- Giữ nguyên các phần tử từ 1 đến k-1.

- Dùng lệnh for gán các giá trị từ a[n+1] đến a[k+1]:

for i:= n+1 downto k+1 n do a[i]:=a[i-1];

- Gán a[k] :=x;

- Thông báo kết quả.

b. - Nhập mảng n phần tử ; nhập vị trí phần tử cần xóa k.

- Giữ nguyên các phần tử từ 1 đến k-1.

- Dùng lệnh for gán các giá trị từ a[k] đến a[n-1]:

for i:= k+1 to n do a[i-1]:=a[i];

- Thông báo kết quả.

7. - Nhập dãy số n số a[i].

- Khởi gán biến dem:=0 để đếm kết quả.

- Dùng lệnh for duyệt tất cả các phần tử của dãy:

Với mỗi i ta lại so sánh a[i] với tất cả các phần tử đứng sau nó nếu xảy ra a[i] trùng với một phần tử nào đó thì bỏ qua nếu không có phần tử nào trùng thì tăng biến dem lên 1, đồng thời ghi phần tử a[i] đó ra màn hình.

- Thông báo kết quả là biến dem.

8. - Khai báo biến mang gồm 6 phần tử kiểu Word;

Type mang=array[1..6] of word;

- Khai báo hằng m kiểu mảng:

Const m:mang=(1,2,5,10,20,50,100);

- Khai báo biến t để nhập số tiền, biến i kiểu số nguyên điều khiển lệnh for.

- Khai báo biến k kiểu mảng để nhập số tờ của từng loại tiền có, biến dem kiểu mảng để ghi số tờ của từng loại tiền.

- Nhập số tiền t;

- Nhập số tờ các loại tiền có vào mảng k và tính tổng số tiền có được.

- Nếu tổng số tiền < t thì thông báo không đủ tiền trả và dừng..

Ngược lại:

Dùng lệnh for dạng lùi duyệt i từ 6 đến 1:

Gán dem[i]:=0;

Lặp lại chừng nào t>=m[i] và k[i] >0 các lệnh:

+ t:=t-m[i];

+ dem[i]:=dem[i]+1;

+ k[i]:=k[i] -1;

Nếu t=0 thì thông báo số tờ từng loại tiền thông qua mảng dem.

Ngược lại thông báo không có đủ tiền lẻ để trả.

9. - Nhập vào ma trận vuông a[i,j] cấp n.

- Các phần tử nằm trên đường chéo chính là a[i,i] (i=1..n).

- Các phần tử nằm phía trên đường chéo chính là a[i,j] (i=1..n, j>i)..

- Khai báo các biến s1,s2 để tính tổng, khởi gán s1:=0; s2:=0; sử dụng lệnh for để tính.

- Thông báo kết quả.

10. - Nhập vào ma trận a[i,j] cấp mxn.

- Nhập k nguyên.

- Dùng 2 lệnh for lồng nhau để duyệt tất cả các phần tử của ma trận nếu thỏa i-j =k thì ghi ra màn hình.

11. - Khởi tạo thủ tục randomize;

- Sử dụng lệnh for và hàm random(9) để lần lượt tạo các chữ số ngẫu nhiên cho từng giải và thông báo ra màn hình..

12. - Nhập xâu st.

- Dùng lệnh while st[1] =#32 do delete(st,1,1) để xóa các ký tự trắng đầu xâu.

- Dùng lệnh while st[length(st)] =#32 do delete(st,length(st),1) để xóa các ký tự trắng cuối xâu.

- Dùng lệnh while pos(#32#32,st) >0 do delete(st, pos(#32#32,st),1) để xóa các ký tự trắng đi liền nhau ở giữa xâu.

13. - Nhập xâu st.

a. Dùng lệnh for duyệt lần lượt các ký tự của xâu (từ 1 đến length(st)), nếu gặp ký tự là chữ hoa thì đổi thành chữ thường bằng cách tăng mã ascii của nó lên 32.

b. Gán thêm ký tự trắng vào đầu xâu st;

- Khởi gán biến dem :=0; để đếm số từ.

- Sử dụng lệnh for để duyệt lần lượt từ 1 đến length(st) - 1. Nếu gặp st[i] =#32 đồng thời st[i+1] <> #32 thì tăng biến đếm lên 1.

- Thông báo kết quả số từ là dem.

14 - Nhập xâu họ và tên cho biến st.

- Xóa ký tự trắng dư thừa ở cuối xâu st.

- Gán i:=length(st);

- Dùng lệnh while duyệt từ sau tới chừng nào mà st[i] còn khác ký tự trắng thì giảm i đi 1 cho đến khi gặp ký tự st[j] khác ký tự trắng đầu tiên thì dừng lại và thông báo tên gồm các ký tự từ vị trí j+1 đến hết xâu.

15 - Nhập số nguyên n

- Khởi gán biến s:=" kiểu xâu để chứa dãy nhị phân.

- Lặp lại công việc sau chừng nào n > 0:

+ Nếu n chẵn thì: s:='0' +s; ngược lại n lẻ thì s:='1' +s;

+ Gán lại n:= n div 2;

- Thông báo kết quả s.

16 - Nhập xâu st bất kỳ.

- Khởi gán biến i:=1; biến count để đếm tổng số từ và khởi gán count:=0;

- Lặp lại công việc sau chừng nào i<= length(st) :

+ Lặp lại lệnh i:=i+1 chừng nào st[i] = ";(Bỏ qua các ký tự trắng).

+ Nếu i <= length(st) thì:

- Chừng nào st[i] <>" thì write(st[i]) và tăng i lên 1 (ghi từ đó ra màn hình). - Tăng biến count lên 1.

- Xuống dòng.

+ Tăng i lên 1.

- Thông báo kết quả số từ là count.

17 - Khai báo các biến so, so1,so2,nho, i, k kiểu nguyên để chứa các số và biến điều khiển lệnh lặp.

- Khai báo các biến s1,s2,ch1,ch2,t,s kiểu xâu.

- Nhập số nguyên thứ nhất lưu vào xâu s1, số nguyên thứ hai lưu vào xâu s2.

- Khởi gán s:='' ;(dùng để lưu kết quả phép cộng hai số); biến nho:=0;(biến nhớ trong phép cộng).

- Nếu độ dài hai xâu s1 và s2 khác nhau thì bổ sung các ký tự '0' vào trước xâu ngắn để được hai xâu có độ dài bằng nhau.

- Dùng lệnh for dạng lùi duyệt từ length(s1) đến 1 và thực hiện phép cộng với các cặp ký tự số tương ứng bằng cách:

    - Dùng thủ tục val đổi ký tự s1[i] thành số so1; ký tự s2[i] thành so2;
    - Gán so:= so1+so2+nho;
    - So:= so mod 10;
    - Nho:= so div 10;
    - Dùng thủ tục str đổi so thành xâu và gán vào biến t;
    - Gán S:=t+S;

- Nếu nho>0 thì gán s:='1'+s;

- Thông báo kết quả là s.

CHƯƠNG 6. CHƯƠNG TRÌNH CON

Mở đầu:

- Trong Turbo Pascal có 2 loại chương trình con: Thủ tục và hàm. Ưu điểm của việc dùng chương trình con là tránh được việc lặp lại một đoạn chương trình nhiều lần trong chương trình, hỗ trợ việc thực hiện các chương trình lớn và phức tạp, phục vụ cho quá trình trừu tượng hóa của các ngôn ngữ lập trình có cấu trúc.

- Chương này đưa ra cách định nghĩa một chương trình con dạng thủ tục, chương trình con dạng hàm. Đồng thời hướng dẫn cách sử dụng các chương trình con này trong một chương trình sao cho phù hợp.

Mục tiêu: Học xong chương này Sinh viên

- Hiểu được thế nào là một chương trình con và các loại chương trình con.

- Biết cách khai báo một chương trình con kiểu thủ tục hay hàm.

- Hiểu được các khái niệm: Tham số hình thức, tham số thực tế, tham trị, tham biến, biến toàn cục, biến địa phương. Biết cách khai báo và sử dụng chúng như thế nào cho phù hợp.

6.1. Các khái niệm mở đầu.

6.1.1. Lập trình từ trên xuống:

Khi lập trình giải quyết một bài toán lớn, nếu viết chương trình một mạch từ trên xuống thì người lập trình sẽ rất vất vả và thậm chí có thể không thực hiện được. Mặc khác, nếu có viết được đi nữa thì việc sửa đổi và kiểm tra lỗi cũng rất là khó khăn vì phải đụng chạm đến toàn bộ chương trình.

Để khắc phục được điều đó, người ta phân tích việc giải quyết bài toán thành những công việc tương đối độc lập. Sau đó chi tiết dần các công việc này thành các các công việc nhỏ hơn. Điều đó có nghĩa là chúng ta đã "phân mảnh" dần theo từng cấp. Turbo Pascal cho phép người lập trình "phân mảnh" chương trình nhằm phục vụ ý đồ trên. Có nghĩa là chia chương trình lớn thành nhiều phần nhỏ rồi giải quyết từng phần một (mỗi phần nhỏ như vậy được gọi là một chương trình con). Sau đó tuỳ nội dung công việc mà lắp ghép các chương trình nhỏ này lại với nhau để giải quyết bài toán.

Phương pháp lập trình có phân mảnh như trên được gọi là TOP-DOWN PROGRAMMING (Lập trình từ gốc đến ngọn hay còn gọi là lập trình có cấu trúc).

6.1.2. Phân loại và cấu trúc chung của chương trình con:

Turbo Pascal cho phép xây dựng hai loại chương trình con đó là thủ tục (Procedure) và hàm (Function). Trong đó:

Thủ tục là một chương trình con dùng để thực hiện một số thao tác xử lý nào đó để giải quyết một công việc cụ thể nào đó đã được phân mảnh.

Hàm là một chương trình con dùng để xác định một giá trị của đại lượng ra nào đó có kiểu dữ liệu đơn giản (số, kí tự, xâu kí tự, logíc). Giá trị của đại lượng ra này được gọi là giá trị trả về của hàm

Việc phân biệt này chỉ có tính cách tương đối nhằm phục vụ cho sự lựa chọn của người lập trình. Mỗi chương trình con đều có cấu trúc chung theo tuần tự như sau:

Phần khai báo chương trình con thuộc loại thủ tục hay hàm.

Các định nghĩa và khai báo địa phương (hằng, kiểu, biến, chương trình con).

Thân của chương trình con.

6.2. Thủ tục.

6.2.1. Định nghĩa thủ tục:

Một thủ tục được định nghĩa theo cú pháp sau:

```
PROCEDURE TênThủTục [(Danh sách các tham số hình thức)];
          Các định nghĩa và khai báo địa phương;
     BEGIN
          Các câu lệnh xử lý; {thân của thủ tục}
     END;
```

Trong đó:

TênThủTục là là một định danh do người sử dụng đặt theo nguyên tắc đặt tên của Turbo Pascal.

Danh sách các tham số hình thức là tên của các đối tượng đóng vai trò nhận thông tin vào cho thủ tục hoạt động. Tham số hình thức có hai loại: tham biến và tham trị. Các tham số hình thức đều được xác định rõ kiểu dữ liệu của chúng và nếu là kiểu dữ liệu của người dùng thì kiểu này phải được định nghĩa trước đó chứ không thể định nghĩa trực tiếp. Nếu là tham biến thì phải có từ khoá `Var` đứng trước.

Ví dụ:

```
PROCEDURE  UCLN(m,n:Word; Var a:Word);
 TYPE  MANG = Array[1..100] Of Integer;
     ........;
PROCEDURE  SapXep(Var A:Mang; N:Word);
     ........;
```

Một thủ tục có thể không có tham số hình thức nào. Khi đó phần khai báo sau tên thủ tục sẽ không có cặp dấu ngoặc.

Ví dụ: `PROCEDURE  TO_MAU;`

Các định nghĩa và khai báo địa phương là các khai báo về hằng, kiểu, biến, chương trình con của nó. Các đối tượng được khai báo ở đây chỉ được dùng cho thủ tục này và các chương trình con chứa trong nó.

Các câu lệnh xử lý là hệ thống câu lệnh được cài đặt nhằm thực hiện giải thuật đã thiết kế cho thủ thục.

Ví dụ:

Để viết thủ tục tìm ước số chung lớn nhất của hai số nguyên dương m và n cho trước ta cho m, n đóng vai trò tham trị, giá trị ước số chung lớn nhất tìm được sẽ lưu ở tham biến a. Thủ tục được viết như sau:

```
PROCEDURE  UCLN(m,n:Word; Var a:Word);
  BEGIN
     If m>n Then a:=n Else a:=m;
     While (m mod a <> 0) Or (n mod a <> 0) Do a:=a-1;
END;
PROCEDURE  TO_MAU;
  Var x1,y1,x2,y2,tg:Byte; {Các khai báo địa phương}
  BEGIN
     Randomize;
     x1:=Random(80)+1; y1:=Random(25)+1;
     x2:=Random(80)+1; y2:=Random(25)+1;
     If x1>x2 Then
       Begin
          tg:=x1;
          x1:=x2;
          x2:=tg;
       End;
     If y1>y2 Then
       Begin
          tg:=y1;
          y1:=y2;
          y2:=tg;
       End;
  Window(x1,y1,x2,y2);
  TextBackGround(Random(16));
  ClrScr;
END;
```

6.2.2. Lời gọi thủ tục:

Sau khi thủ tục đã được định nghĩa, muốn sử dụng thủ tục đó thì phải thực hiện lời gọi thủ tục. Lời gọi thủ tục được thực hiện như sau:

- Nếu thủ tục có tham số hình thức thì lời gọi thủ tục phải có tên thủ tục và các tham số thực tế kèm theo như sau:

```
.....;
TênThủTục [(danh sách tham số thực tế)];
.....;
```

- Nếu thủ tục không có tham số hình thức thì lời gọi thủ tục chỉ có tên thủ tục:

```
..... ;
TênThủTục ;
..... ;
```

Nếu định nghĩa thủ tục có bao nhiêu tham số hình thức thì lời gọi thủ tục phải có đử bấy nhiêu tham số thực tế. Các tham số thực tế theo thứ tự sẽ lần lượt thay thế cho các tham số hình thức, các tham số thực tế được viết phân cách nhau bởi một dấu phẩy. Tham số thực tế thay thế cho tham số hình thức nào thì phải có đúng kiểu dữ liệu của tham số hình thức đã khai báo. Ngoài ra cần chú ý:

Tham số thực tế thay cho tham trị có thể là hằng, biến, biểu thức hay lời gọi hàm (vì đây là các đại lượng có giá trị xác định đóng vai trò nhận thông tin vào cho thủ tục hoạt động).

Tham số thực tế thay cho tham biến bắt buộc phải là biến đã được khai báo trong phạm vi thủ tục có thể sử dụng được (vì chỉ có biến được khai báo ngoài thủ tục mới có thể lưu giữ thông tin ra).

Khi thực hiện lời gọi thủ tục, nếu có truyền biến cho tham biến hoặc tham trị thì trong quá trình xử lý giá trị của biến có thể bị thay đổi. Tuy nhiên, khi thủ tục hoạt động xong thì:

Nếu tham số thực tế là biến truyền theo kiểu tham trị thì biến đó sẽ lấy lại giá trị của nó trước khi truyền cho thủ tục.

Nếu tham số thực tế là biến truyền theo kiểu tham biến thì biến đó sẽ giữ lại giá trị của lần thay đổi sau cùng.

Ngoài ra, Turbo Pascal cho phép trong thân các chương trình con có thể có lời gọi đến các chương trình con đã được định nghĩa trước nó.

Ví dụ:

```
USES CRT;
     Var m,n:Word;
  PROCEDURE  Wait;
       Begin
          Writeln('go phim bat ky de tiep tuc');
          Repeat  Until  Keypressed;
       End;
  PROCEDURE  P1(m:Word; Var n:Word);
       Begin
          If m>n Then m:=m-n Else m:=n-m;
          n:=2*m;
          Writeln('m=',m,'n=',n);
       End;
BEGIN
     m:=5; n:=8;                    {1}
```

```
        Writeln('m=',m,'n=',n);        {2}
        P1(m,n);                       {3}
        Wait;                          {4}
        Writeln('m=',m,'n=',n);        {5}
        Wait;                          {6}
END.
```

Khi chạy chương trình trên thì kết quả thực hiện của các dòng lệnh được giải thích như sau:

{1} Gán giá trị cho biến m và n, ta được m=5 và n=8.

{2} Viết giá trị của biến m=5 và n=8 ra màn hình.

{3} Thực hiện lời gọi thủ tục P1. Trong thủ tục nầy có sự thay đổi giá trị của m và n, vì m>n có giá trị False nên thực hiện phép gán m:=n-m, tức là m=3, sau đó thực hiện phép gán n:=2*m, tức là n=6. Cuối cùng viết giá trị của m=3 và n=6 tại thời điểm nầy ra màn hình.

Khi kết thúc thủ tục nầy, vì m truyền theo kiểu tham trị và n truyền theo kiểu tham biến nên m lấy lại giá trị ban đầu của nó, tức m=5 và n lưu giữ giá trị của lần thay đổi sau cùng, tức n=6.

{4} Thực hiện thủ tục Wait.

{5} Viết giá trị của m và n tại thời điểm nầy ra màn hình, tức m=5 và n=6.

{6} Thực hiện thủ tục Wait.

Như vậy, khi thực hiện chương trình trên sẽ cho ta kết quả in lên màn hình như sau:

```
m = 5    n = 8
m = 3    n = 6
go phim bat ky de tiep tuc
m = 5    n = 6
go phim bat ky de tiep tuc
```

6.3. Biến toàn cục và biến địa phương.

6.3.1. Định nghĩa:

Biến toàn cục là các biến được khai báo sau từ khoá `VAR` của chương trình chính, còn ***biến địa phương*** là các biến được khai báo sau từ khoá `VAR` trong các chương trình con.

```
USES CRT;
VAR  Khai báo các biến toàn cục;
PROCEDURE  AAA(danh sách các tham số hình thức);
```

```
  Var Khai báo các biến địa phương;
     Begin
          ........ ;
     End;

BEGIN
     ........ ;
END.
```

Phạm vi sử dụng của biến địa phương là trong thân của chương trình con khai báo chúng và trong các chương trình con chứa trong chương trình con này. Thời gian tồn tại của chúng là từ khi chương trình con được gọi thực hiện cho đến khi thực hiện xong. Còn phạm vi sử dụng của biến toàn cục là trong toàn bộ chương trình (trong chương trình chính và trong tất cả các chương trình con của nó) và thời gian tồn tại của chúng là khi chương trình đang hoạt động.

6.3.2. Chú ý:

Tên của các tham số hình thức và tên của các biến địa phương trong cùng một chương trình con không được trùng nhau.

Tên của các biến địa phương có thể trùng tên với tên biến toàn cục. Tuy nhiên khi chương trình con được gọi, nếu có sử dụng biến trùng tên thì trong quá trình chương trình con hoạt động, Turbo Pascal sẽ hiểu đó là biến địa phương, khi đó biến toàn cục tạm thời bị che dấu cho đến khi chương trình con hoạt động xong.

Biến toàn cục có thể tham gia trong các chương trình con, mọi tác động ảnh hưởng đến nó vẫn giống như khi chúng tham gia trong chương trình chính. Ngược lại, biến địa phương chỉ tham gia trong chương trình con khai báo chúng, không thể tham gia trong chương trình chính.

Các ví dụ:

1.
```
Uses  Crt;
Var a,b,c:Word;
PROCEDURE  P1(m:Word; Var n:Word);
  Begin
     If m>n Then m:=m-n Else m:=n-m;
     n:=2*m;
  End;
BEGIN
     a:=5; b:=8; c:=10;                {1}
     P1(b+c,a);                        {2}
     Writeln('a=',a,'b=',b,'c=',c);    {3}
END.
```

Các bước thực hiện của chương trình được giải thích như sau:

{1} Gán giá trị cho các biến toàn cục a, b, c.

{2} Thực hiện lời gọi thủ tục P1, vì m=b+c=8+10=18 và n=a=5 nên m>n do đó phép gán m:=m-n, tức là m:=13 được thực hiện. Sau đó phép gán n:=2*m được thực hiện, tức là n = 26. Sau khi kết thúc, vì a được truyền theo kiểu tham biến cho tham số hình thức n nên a lưu giữ giá trị của lần thay đổi sau cùng là 26.

Như vậy, khi thực hiện chương trình trên sẽ cho ta kết quả in lên màn hình như sau: a = 26 b = 8 c = 10

2.

```
Uses Crt;
Var a,b,c:Word;
PROCEDURE  P1(m:Word; Var n:Word);
  Begin
     If m>n Then m:=m-n Else m:=n-m;
     n:=2*m;
     a:=m+n;   {phép gán này ở ví dụ 1 không có}
  End;
BEGIN
     a:=5; b:=8; c:=10;                {1}
     P1(b+c,a);                        {2}
     Writeln('a=',a,'b=',b,'c=',c);    {3}
END.
```

Như vậy, khi thực hiện chương trình trên sẽ cho ta kết quả in lên màn hình như sau: a = 39 b = 8 c = 10

Vấn đề khác ở trên là do có tác động phép gán a:=m+n, tức là a=13+26=39. Vì a là biến toàn cục (không trùng với tên tham số hình thức và tên biến địa phương nào) nên mọi tác động thay đổi giá trị trên nó đều có ý nghĩa.

3.

```
Uses Crt;
Var a,b,c:Word;
PROCEDURE  P1(m,n:Word);
  Var a:Word;
  Begin
     If m>n Then a:=m Else a:=n;
     Writeln('a=',a,'b=',b,'c=',c);
  End;
BEGIN
     a:=5; b:=8; c:=10;                {1}
     P1(b,c);                          {2}
     Writeln('a=',a,'b=',b,'c=',c);    {3}
END.
```

Các bước thực hiện của chương trình được giải thích như sau:

{1} Gán giá trị cho các biến toàn cục a, b, c.

{2} Thực hiện lời gọi thủ tục P1 với m = b = 8 và n = c = 10. Trong thủ tục này có biến địa phương a trùng tên với biến toàn cục. Vì m>n có giá trị False nên phép gán a:=n được thực hiện, tức là a:=10 được thực hiện (chú ý ở đây a là biến địa phương). Sau đó viết ra màn hình giá trị của a (biến địa phương) và b, c (biến toàn cục).

Như vậy, khi thực hiện chương trình trên sẽ cho ta kết quả in lên màn hình như sau:

a = 10 b = 8 c = 10

a = 5 b = 8 c = 10

Trong đó dòng đầu được viết bởi câu lệnh Writeln trong thủ tục P1 với a là biến địa phương và dòng sau được viết bởi câu lệnh Writeln trong chương trình chính với a là biến toàn cục.

6.4. Hàm.

6.4.1. Định nghĩa hàm:

Một hàm được định nghĩa theo cú pháp sau:

```
FUNCTION  TênHàm [(Danh sách các tham số hình thức)] :<Kiểu>;
          Các khai báo địa phương;
          Định nghĩa các chương trình con của hàm;
     BEGIN
          Các câu lệnh xử lý; {thân của hàm}
     END;
```

Trong đó:

TênHàm là một định danh do người sử dụng đặt theo nguyên tắc đặt tên của Turbo Pascal.

Danh sách các tham số hình thức giống như đối với thủ tục.

Kiểu là kiểu của giá trị trả về của hàm.

Ví dụ: `FUNCTION UCLN(m,n:Word):Word;`

Một hàm có thể không có tham số hình thức nào. Khi đó phần khai báo sau tên hàm sẽ không có cặp dấu ngoặc.

Ví dụ: `FUNCTION SO_NGAU_NHIEN:Word;`

Các khai báo địa phương giống như đối với thủ tục.

Các câu lệnh xử lý giống như đối với thủ tục. Tuy nhiên đối với hàm trong thân hàm bắt buộc phải có câu lệnh gán tên hàm bằng giá trị trả về của hàm, giá trị này có thể dưới dạng một biểu thức có kiểu là kiểu của giá trị trả về của hàm, theo cú pháp:

TênHàm := Biểu thức; {giá trị của biểu thức là giá trị trả về của hàm}

Trong đó kiểu của Biểu thức phải trùng với kiểu của giá trị trả về của hàm.

6.4.2. Lời gọi hàm:

Khi sử dụng hàm thì phải thực hiện lời gọi hàm. Thực hiện lời gọi hàm là để lấy giá trị trả về của hàm. Vì vậy lời gọi hàm phải được đặt trong một biểu thức có xử lý giá trị của nó. Khi thực hiện lời gọi hàm thì cũng phải truyền tham số thực tế cho nó giống như đối với thủ tục.

Ví dụ:

```
USES  CRT;
Var m,n,a:Word;
FUNCTION  SO_NGAU_NHIEN:Word;
  BEGIN
     SO_NGAU_NHIEN:=Random(100)+1;
  END;
FUNCTION  UCLN(m,n:Word):Word;
  Var a:Word;
  BEGIN
     If m>n Then a:=n Else a:=m;
     While (m mod a <> 0) Or (n mod a <> 0) Do a:=a-1;
     UCLN:=a;
  END;
BEGIN
  Randomize;
  m:=SO_NGAU_NHIEN; n:=SO_NGAU_NHIEN;
  Writeln('USCLN cua ', m,' va ', n,' la: ', UCLN(m,n))
END.
```

Chương trình trên gồm có hai hàm:

Hàm SO_NGAU_NHIEN có giá trị trả về của hàm là lấy ngẫu nhiên một số nguyên dương bé hơn hoặc bằng 100.

Hàm UCLN(m,n) có giá trị trả về của hàm là ước số chung lớn nhất của hai số nguyên dương m và n.

(độc giả tự tìm hiểu các giải thuật đã sử dụng trong các hàm trên).

6.5. Chương trình con lồng nhau.

Trong một chương trình con, ở phần các khai báo địa phương, ta có thể định nghĩa các chương trình con phục vụ riêng cho nó. Và việc này có thể tiếp tục được thực hiện thêm theo nhiều mức của các chương trình con lồng nhau. Ta có thể minh hoạ như sau:

```
PROCEDURE  P1(các tham số hình thức);
FUNCTION  F1(các tham số hình thức):Word;
  Begin
```

```
      {Thân của chương trình con F1}
  End;
PROCEDURE  P2(các tham số hình thức);
PROCEDURE  P3(các tham số hình thức);
  Begin
   {Thân của chương trình con P3}
  End;
Begin
  {Thân của chương trình con P2}
End;
Begin
  {Thân của chương trình con P1}
End;
```

Theo cấu trúc như trên, thì:

F1, P2 là các chương trình con được định nghĩa trong chương trình con P1. Như vậy trong thân của chương trình con P1 có thể có lời gọi đến các chương trình con F1, P2 và dĩ nhiên không thể gọi chương trình con P3 được.

P3 là chương trình con được định nghĩa trong chương trình con P2. Như vậy trong thân của chương trình con P2 có thể có lời gọi đến chương trình con P3.

6.6. Chương trình con đệ qui.

6.6.1. Thế nào là một chương trình con đệ qui?

Chương trình con đệ qui là một chương trình con mà trong thân của nó có ít nhất một câu lệnh là lời gọi đến chính nó. Trong từng trường hợp cụ thể là thủ tục hay hàm mà ta gọi đó là thủ tục đệ qui hay hàm đệ qui.

6.6.2. Các ví dụ:

Viết hàm đệ qui để tính tổng S = 1+2+... + n với n là số nguyên dương cho trước.

```
FUNCTION  S(n:Word):Word;
     Begin
          If n = 1 Then S := 1
          Else S := S(n-1)+n ;
     End;
```

Viết hàm đệ qui để xac định ước số chung lớn nhất của hai số nguyên dương m và n cho trước:

```
FUNCTION  UCLN(m,n:Word):Word;
  Begin
     If m mod n = 0 Then UCLN := n
     Else UCLN := UCLN(n,m mod n);
  End;
```

Viết thủ tục đệ qui để viết ra màn hình số dạng nhị phân của một số nguyên dương n cho trước:

```
Procedure  NhiPhan(n:Word);
  Begin
     If n div 2 <> 0 Then NhiPhan(n div 2);
     Write(n mod 2);
  End;
```

(Độc giả tự tìm hiểu các giải thuật đã được sử dụng cho các chương trình con đệ qui nói trên)

***Nhận xét*:**

- Việc viết một chương trình con đệ qui thường rất là ngắn gọn và được áp dụng trong một số tình huống rất tốt. Tuy nhiên cần chú ý các điểm sau:

- Khi thực hiện lời gọi đệ qui (tức lời gọi đến chính nó) thì Turbo Pascal lưu giữ các giá trị của các biến địa phương và tham số vào Stack, mà Stack chỉ được cấp phát với kích thước không quá 64 KB, nên có thể dẫn đến tình trạng tràn Stack (Stack Overflow).

- Turbo Pascal cho phép ta có thể mở rộng tối đa vùng sử dụng của Stack bằng cách dùng dẫn biên dịch {$M 65000,0,655360}.

- Để khắc phục tình trạng tràn Stack, thông thường người lập trình tổ chức hạn chế tối đa sử dụng biến địa phương và số câu lệnh đặt sau lời gọi đệ qui, hoặc thay thế bởi một chương trình con không đệ qui (gọi là khử đệ qui) bằng cách sử dụng các câu lệnh lặp thay thế.

- Phải chú ý trường hợp suy biến để chương trình đệ qui dừng (tức là đến một lúc nào đó sẽ không thực hiện lời gọi đệ qui nữa).

6.6.3. Đệ qui gián tiếp:

Như đã trình bày ở trên, một chương trình con chỉ có thể thực hiện lời gọi một chương trình con đã được định nghĩa trước nó, tức là không thể gọi chương trình con định nghĩa sau nó. Tuy nhiên, trong thực tế có thể xảy ra điều này. Vì vậy Turbo Pascal cho phép thực hiện được điều trên bằng cách dùng từ khoá FORWARD theo cú pháp như sau:

```
..... ;
PROCEDURE/FUNCTION P1[(các tham số hình thức)]/:<Kiểu>;
FORWARD ; {P1 được khai báo trước nhưng chưa định nghĩa}
PROCEDURE/FUNCTION P2[(các tham số hình thức)]/:<Kiểu>;
     Các khai báo địa phương;
     Begin
          ..... ;
          { Có sử dụng chương trình con P1 }
          ..... ;
```

```
     End;
PROCEDURE/FUNCTION  P1[(các tham số hình thức)]/:<Kiểu>;
     Các khai báo địa phương;
     Begin
          .......... ;
          { Có sử dụng chương trình con P2 }
          .......... ;
     End;
     .......... ;
```

Về mặt cú pháp thì P1 được khai báo trước với từ khoá FORWARD kèm theo (chưa định nghĩa P1), sau đó định nghĩa P2 (có gọi P1) và tiếp liền sau phải định nghĩa P1 (có gọi P2).

Như vậy đó cũng là một hình thức đệ qui mà ta gọi là đệ qui gián tiếp. Ngược lại nếu có lời gọi trực tiếp đến chính nó như trước đây thì ta gọi là đệ qui trực tiếp.

6.7. Đơn vị chương trình của người dùng.

6.7.1. Đơn vị chương trình (UNIT) là gì?

Để phục vụ cho người lập trình, Turbo Pascal đã biên soạn sẵn một số chương trình con, gọi là chương trình con chuẩn nhằm trợ giúp một số thao tác dữ liệu. Các chương trình con này được đặt trong file TURBO.TPL. Các chương trình con chuẩn này được phân chia và đặt trong các phần riêng, mỗi phần được gọi là một đơn vị chương trình chuẩn. Như vậy, các đơn vị chương trình chuẩn chứa các thủ tục chuẩn, hàm chuẩn mà chúng ta đã có dịp sử dụng trong khi lập trình. Chẳng hạn như các thủ tục Write(....), Readln(....), Gotoxy(....),.... và các hàm chuẩn Abs(....), Sin(....), Keypressed,....

Turbo Pascal còn cho phép người sử dụng xây dựng các đơn vị chương trình riêng của mình, gọi là đơn vị chương trình của người dùng.

Khi cần sử dụng một chương trình con nào thì phải biết nó được chứa trong đơn vị chương trình nào (đơn vị chương trình chuẩn hay của người dùng) và phải khai báo đơn vị chương trình đó sau từ khoá USES ngay đầu chương trình, ngoại trừ các chương trình con chứa trong đơn vị chương trình chuẩn SYSTEM.

6.7.2. Các đơn vị chương trình chuẩn của Turbo Pascal:

Turbo Pascal có các đơn vị chương trình chuẩn sau đây:

SYSTEM là đơn vị chương trình chứa các hằng, biến, hàm, thủ tục thường dùng như xuất nhập dữ liệu, cấp phát và quản lý bộ nhớ. Đơn vị chương trình này sẽ được nộp thường trực vào bộ nhớ khi Turbo Pascal được khởi động.

Sau đây là một số thủ tục và hàm chuẩn thường dùng chứa trong đơn vị chương trình SYSTEM:

Thủ tục:

Append	Mở tập tin dạng văn bản để ghi thêm vào cuối.
Assign	Gán tên tập tin cho biến file.
Close	Đóng tập tin.
Delete	Xóa một chuỗi kí tự con.
Dispose	Giải phóng biến động.
Erase	Xóa tập tin.
Inset	Chèn một xâu con vào một xâu khác.
Mark	Đánh dấu các biến động.
New	Cấp phát biến động.
Randomize	Khởi tạo chế độ ngẫu nhiên.
Read, Readln	Gán giá trị nhập từ bàn phím của một hay nhiều biến.
Release	Giải phóng các biến động đã bị đánh dấu.
Rename	Đổi tên tập tin.
Reset	Mở tập tin đã có.
Rewrite	Tạo và mở tập tin mới.
Str	Chuyển đổi dạng số thành dạng xâu.
Seek	Chuyển vị trí cửa sổ file của tập tin dạng định kiểu.
Val	Chuyển đổi dạng chuỗi thành số.
Write, Writeln	Xuất dữ liệu ra màn hình hay tập tin.

Hàm:

Abs	Lấy giá trị tuyệt đối.
Arctan	Lấy giá trị arctang của một giá trị lượng giác.
Copy	Lấy xâu con của một xâu.
Cos	Lấy giá trị cosin của một giá trị lượng giác.
Concat	Lấy kết quả nối hai xâu.
Eof	Lấy trạng thái kết thúc file của một tập tin.
Eoln	Lấy trạng thái kết thúc dòng của một biến tập tin dạng văn bản.
Exp	Lấy lũy thừa cơ số e của một số.
FilePos	Lấy vị trí của sổ file trong tập tin dạng định kiểu.
FileSize	Lấy kích thước của tập tin dạng định kiểu.
Frac	Lấy phần thập phân của một số thực.
Int	Lấy phần nguyên của một số thực.
IOResult	Lấy kết quả mã lỗi của việc mở một tập tin bởi thủ tục Reset.
Ln	Lấy giá trị logarit tự nhiên của một số.

Ord	Lấy mã ASCII của một kí tự.
Pi	Giá trị 3.1416...
Pos	Lấy vị trí của xâu con trong một xâu khác.
Pred	Lấy giá trị đứng trước đối tượng có kiểu: nguyên, kí tự, boolean.
Random	Lấy một số nguyên ngẫu nhiên.
Round	Lấy giá trị nguyên gần số thực nhất.
Sin	Lấy giá trị sin của một giá trị lượng giác.
Sqr	Lấy giá trị bình phương của một số.
Sqrt	Lấy giá trị căn bậc hai của một số dương.
Succ	Lấy giá trị đứng sau đối tượng có kiểu: nguyên, kí tự, boolean
Trunc	Lấy phần nguyên của một số thực.
Upcase	Lấy kí tự in hoa của một kí tự in thường.

CRT là đơn vị chương trình chứa các hằng, biến, hàm và thủ tục liên quan đến xử lý âm thanh, xuất dữ liệu dạng văn bản trên màn hình và nhập dữ liệu từ bàn phím. Khi sử dụng các chương trình con trong đơn vị chương trình này thì phải khai báo CRT sau từ khoá USES.

Sau đây là một số hằng, biến, thủ tục và hàm chuẩn thường dùng chứa trong đơn vị chương trình CRT:

Các hằng liên quan đến chế độ phân giải của màn hình.

BW40 = 0 ; CO40 = 1 ; BW80 = 2 ; CO80 = 3 ;

Mono = 7 ; Font8x8 = 256 ;

Các hằng liên quan đến chế độ màu của màn hình

Black	= 0	LightBlue	= 9
Blue	= 1	LightGreen	= 10
Green	= 2	LightCyan	= 11
Cyan	= 3	LightRed	= 12
Red	= 4	LightMagenta	= 13
Magenta	= 5	Yellow	= 14
Brown	= 6	White	= 15
LightGray	= 7	Blink	= 128
DarkGray	= 8		

Các thủ tục

ClrEol	Xóa trên màn hình từ vị trí con trỏ đến cuối dòng.
ClrScr	Xóa cửa số màn hình hiện tại.

Delay Dừng chương trình một thời gian.

DelLine Xóa dòng tại vị trí con trỏ.

GotoXY Đưa con trỏ đến một vị trí tọa độ màn hình.

InsLine Chèn một dòng trắng trên màn hình tại vị trí con trỏ.

NoSound Tắt âm thanh.

Sound Phát âm thanh với một tần số.

TextBackGround Xác định màu nền.

TextColor Xác định mà của ký tự.

Window Mở (xác định) cửa sổ màn hình.

Các hàm

KeyPressed Lấy trạng thái của việc bấm một phím nào đó hay chưa.

Readkey Lấy giá trị của một phím được bấm (không cần gõ Enter)

WhereX Lấy tọa độ cột của con trỏ trên màn hình.

WhereY Lấy tọa độ dòng của con trỏ trên màn hình.

DOS chứa các hằng, biến, thủ tục và hàm chuẩn liên quan đến hệ thống (truy xuất tập tin trên đĩa, truy xuất bộ nhớ, các vectơ ngắt, gọi các chương trình ngoài và chương trình thường trú). Khi sử dụng các chương trình con trong đơn vị chương trình này thì phải khai báo DOS sau từ khoá USES.

Sau đây là một số thủ tục và hàm chuẩn thường dùng chứa trong đơn vị chương trình DOS:

Thủ tục liên quan đến ngày, giờ của hệ thống:

GetDate Lấy ngày của hệ thống.

GetTime Lấy giờ của hệ thống.

SetDate Định lại ngày cho hệ thống.

SetTime Định lại giờ cho hệ thống.

Hàm lấy tình trạng đĩa:

DisFree Lấy số byte còn trống trên đĩa.

DiskSize Lấy dung lượng của đĩa.

GRAPH chứa các hằng, biến, thủ tục và hàm chuẩn liên quan đến xuất dữ liệu ra màn hình trong chế độ đồ hoạ. Khi sử dụng các chương trình con chứa trong đơn vị chương trình này thì phải khai báo GRAPH sau từ khoá USES.

Các thủ tục và hàm chuẩn thường dùng chứa trong đơn vị chương trình GRAPH độc giả sẽ được biết trong giáo trình lý thuết đồ họa.

PRINTER chứa các thủ tục và hàm chuẩn liên quan đến máy in. Khi sử dụng các chương trình con trong đơn vị chương trình này thì phải khai báo PRINTER sau từ khoá USES.

Đây là đơn vị chương trình chứa duy nhất một biến dạng tập tin mô tả máy in để chương trình xuất dữ liệu ra máy in. Đó là:

```
Var  Lst : Text ;
```

6.7.3. Đơn vị chương trình của người dùng:

Ngoài ra, Turbo Pascal còn cho phép người lập trình tạo ra các đơn vị chương trình của chính mình, gọi là đơn vị chương trình của người dùng, nhằm chứa các thủ tục và hàm do mình biên soạn và lưu giữ trong một file riêng để thuận tiện cho việc sử dụng chúng nhiều lần.

Việc tạo ra đơn vị chương trình của người sử dụng được tiến hành tuần tự như sau:

Từ môi trường làm việc của Turbo Pascal thực hiện biên soạn chương trình nguồn và ghi lên đĩa với file có tên *.PAS.

Biên dịch ghi lên đĩa, Turbo Pascal tự động ghi nội dung biên dịch được thành file với tên *.TPU, tên của file này sẽ trùng với tên file nguồn chứa đơn vị chương trình được biên dịch và cũng chính là tên của đơn vị chương trình của người dùng.

Cấu trúc chung của chương trình nguồn:

Cấu trúc chung của một đợn vị chương trình có 4 phần:

- Phần khai báo
- Phần giao tiếp (Interface)
- Phần cài đặt (Implementation)
- [Phần khởi tạo (Initialization)]

Phần khai báo gồm từ khoá UNIT và tên của đơn vị chương trình, tên này buộc phải trùng với tên của file lưu chương trình nguồn.

Phần giao tiếp mô tả tất cả các đối tượng (hằng, biến, kiểu, thủ tục, hàm,....) mà các chương trình nào khai báo sử dụng đơn vị chương trình này đều có thể dùng được. Nếu là thủ tục hoặc hàm thì phần khai báo là khai báo phần đầu của chương trình con dạng thủ tục hoặc hàm đó.

Phần cài đặt chứa các khai báo riêng chỉ sử dụng trong nội bộ đơn vị chương trình này và định nghĩa một cách hoàn chỉnh các thủ tục và hàm đã được khai báo ở phần giao tiếp. Khi định nghĩa các thủ tục và hàm thì cần chú ý phải bảo đảm đúng thứ tự của chúng như đã khai báo ở trên (chương trình con nào được khai báo trước thì định nghĩa trước, khai báo sau thì định nghĩa sau). Ngoài ra ở đây còn cho phép định nghĩa thêm một số chương trình con khác chỉ để dùng trong nội bộ của đơn vị chương trình này.

Phần khởi tạo dùng để chứa một số câu lệnh sẽ thực hiện trước nhằm giải quyết một số công việc nào đó trước khi chương trình chính của chương trình có sử dụng đơn vị chương trình này thực hiện. Phần này có thể có hoặc không.

Như vậy, giả sử trên đĩa có có file với tên MYUNIT.PAS chứa chương trình nguồn của một đơn vị chương trình thì nó có dạng như sau:

```
UNIT   MYUNIT;
INTERFACE
Các khai báo USES, CONST, TYPE, VAR;
Các khai báo PROCEDURE, FUNCTION;
IMPLEMENTATION
Các khai báo USES, CONST, TYPE, VAR;
```

Cài đặt (định nghĩa) các thủ tục và hàm đã khai báo ở trên hoặc các chương trình con để sử dụng nội bộ.

```
BEGIN     {nếu không có lệnh khởi tạo nào thì
                    không cần từ khoá BEGIN này}
          Các câu lệnh khởi tạo;
END.
```

Ví dụ:

Chúng ta cần tạo một đơn vị chương trình của người sử dụng có tên MYUNIT để khi sử dụng đơn vị chương trình này ta có thể dùng đơn vị chương trình chuẩn CRT, kiểu dữ liệu của người dùng STR30, các thủ tục và hàm INHOA, BCNN, LietKeSNT do người dùng định nghĩa. Khi đó từ môi trường soạn thảo của Turbo Pascal ta tạo file chứa chương trình nguồn có tên MYUNIT.PAS với nội dung chương trình nguồn được biên soạn như sau:

```
UNIT   MYUNIT;

INTERFACE
  USES  CRT;
  TYPE  STR30 = STRING[30];
  PROCEDURE  INHOA(Var St:STR30);
  FUNCTION   BCNN(m,n:Word):Word;
  PROCEDURE  LietKeSNT(n:Word);
IMPLEMENTATION
  PROCEDURE  INHOA(Var St:STR30);
     Var I:Word;
     Begin
       For I:= To Length(St) Do St[I]:=Upcase(St[I]);
     End;
  FUNCTION   UCLN(m,n:Word):Word;
     Begin
       If m mod n = 0 Then UCLN:=n
       Else UCLN:=UCLN(n, m mod n);
     End;
  FUNCTION   BCNN(m,n:Word):Word;
     Begin
```

```
      BCNN:=(m*n) div UCLN(m,n);
    End;
  FUNCTION   KTraSNT(n:Word):Boolean;
    Var a:Word;
    Begin
      a:=2;
      While (n mod a <> 0) And (a < n div 2) Do a:=a+1;
      If a > n div 2 Then KTraSNT:=True
      Else KTraSNT:=False;
    End;
  PROCEDURE  LietKeSNT(n:Word);
    Var I:Word;
    Begin
      For I:=2 To n Do
      If KTraSNT(I) Then Write(I:6);
    End;
END.
```

Giải thích thêm: Chương trình nào viết bằng Turbo Pascal có sử dụng đơn vị chương trình MYUNIT ở trên thì trong chương trình nguồn của nó:

Có thể sử dụng đơn vị chương trình `CRT`, kiểu `STR30` và các chương trình con `INHOA(..), BCNN(..), LietKeSNT(..),` vì đây là các đối tượng được khai báo sau từ khoá `INTERFACE`.

Không sử dụng được các chương trình con `UCLN(..)` và `KTraSNT(..)` vì đây là các chương trình con chỉ được định nghĩa sau từ khoá `IMPLEMENTATION` nhằm phục vụ cho đơn vị chương trình MYUNIT. Cụ thể hàm `UCLN(..)` để xác định ước chung lớn nhất của hai số được dùng trong hàm `BCNN(..)` xác định bội chung nhỏ nhất của hai số, hàm `KTraSNT(..)` để kiểm tra một số có phải là số nguyên tố hay không được dùng trong thủ tục `LietKeSNT(..)` để liệt kê các số nguyên tố.

Biên dịch đơn vị chương trình thành file dạng *.TPU:

Trong quá trình soạn thảo file chứa chương trình nguồn của đơn vị chương trình ta sử dụng phím F9 để biên dịch lỗi cú pháp (nhưng không thể sử dụng Ctrl-F9 để chạy một đơn vị chương trình). Khi file nguồn đã ổn định (không sai lỗi cú pháp) ta chọn chế độ dịch ghi đĩa và ấn F9, khi đó file nguồn được biên dịch thành file mới dạng mã máy và ghi lên đĩa với tên là tên của file nguồn và phần mở rộng là TPU. Tức là file có tên dạng *.TPU.

Ví dụ với file chứ chương trình nguồn như trên thì được biên dịch thành file có tên MYUNIT.TPU

Sử dụng đơn vị chương trình của người dùng:

Mỗi khi đã có một đơn vị chương trình của người dùng, một chương trình hoặc một đơn vị chương trình nào khác muốn sử dụng các đối tượng đã được khai báo sau từ khoá INTERFACE của đơn vị chương trình nầy thì khai báo tên của nó sau từ khoá USES.

Giả sử đã có đơn vị chương trình với chương trình nguồn lưu ở file MYUNIT.PAS và đã được biên dịch thành file MYUNIT.TPU thì ta có thể sử dụng nó trong một chương trình viết bằng Turbo Pascal như sau:

```
PROGRAM   THU_NGHIEM;
     USES  CRT, MYUNIT;
     {có khai báo đơn vị chương trình MYUNIT}
     Var  s:STR30;
                    a,b:Word;
     BEGIN
          Write('Nhap vao mot xau:'); Readln(s);
          INHOA(s);
          Write('Xau sau khi doi thanh in hoa la:',s);
          Write('Nhap vao 2 so nguyen:'); Readln(a,b);
          Write('Boi so chung nho nhat la:',BCNN(a,b));
     END.
```

Nhận xét:

Khi xây dựng các chương trình con ta cần xác định rỏ dữ liệu vào cùng với kiểu dữ liệu của chúng để tổ chức các tham số hình thức và xác định dữ liệu ra. Dựa vào việc xác định dữ liệu ra để chọn cách thể hiện chương trình con dưới dạng hàm hay thủ tục. Một số nhận xét sau đây là một những gợi ý giúp cho người bắt đầu học lập trình xác định cho mình một cách chọn lựa:

Kiểu của giá trị trả về của hàm phải là kiểu đơn trị (kiểu số, lôgic, kí tự, xâu kí tự, con trỏ), không được đa trị (kiểu tập hợp, mảng, bản ghi, tập tin).

Nếu chỉ yêu cầu một dữ liệu ra, ta có thể chọn chương trình con dạng hàm hoặc thủ tục. Nếu là hàm thì giá trị trả về của hàm là dữ liệu ra đó (ví dụ hàm xác định ước chung lớn nhất của hai số nguyên dương thì giá trị trả về của hàm sẽ là ước chung lớn nhất của hai số nguyên dương đó). Nếu chọn là thủ tục thì phải có một tham biến lưu giữ giá trị sau cùng của dữ liệu ra.

Nếu lấy nhiều hơn một dữ liệu ra, ta cũng có thể chọn chương trình con dạng hàm hay thủ tục. Nếu là hàm thì chọn một dữ liệu ra là giá trị trả về của hàm, số còn lại tổ chức các tham biến để lưu giữ. Nếu là thủ tục thì tổ chức các tham biến lưu giữ chúng.

BÀI TẬP CHƯƠNG 6.

1. Xây dựng hàm log(a,b:Real), trong đó a và b là các số nguyên dương, giá trị trả về của hàm là giá trị của $\log_a b$.

2. Xây dựng hàm Max(a,b:Real) có giá trị trả về của hàm là giá trị lớn nhất trong hai số a và b.

3. Xây dựng hàm BCNN(a,b:Worrd) có giá trị trả về của hàm là bội số chung nhỏ nhất của hai số nguyên dương a và b.

4. Xây dựng hàm DaoNguoc(n:Word) có giá trị trả về của hàm là số nguyên có được bằng cách lấy đảo ngược các chử số của n. (Ví dụ khi thực hiện các lời gọi hàm sau ta sẽ được: DaoNguoc(12305) = 50321; DaoNguoc(12300) = 321).

5. Xây dựng hàm SNT(n:Word) có giá trị trả về của hàm là True nếu n là số nguyên tố, ngược lại thì giá trị trả về của hàm là False.

6. Hãy sử dụng hàm SNT ở trên để viết thủ tục liệt kê lên màn hình tất cả các số nguyên tố bé hơn hoặc bằng n, mỗi dòng có 10 số, ngoại trừ dòng sau cùng, giữa hai số cách nhau ít nhất một kí tự trắng. Trong đó n là số nguyên dương cho trước.

7. Xây dựng hàm S(n:Word), bằng hai cách: đệ qui và không đệ qui, có giá trị trả về của hàm là giá trị của tổng $S = 1^2 + 2^2 + 3^2 + ... + n^2$.

8. Xây dựng hàm S(n:Word;x:Real), bằng hai cách: đệ qui và không đệ qui, có giá trị trả về của hàm là giá trị của tổng $S = x\sin x + x^2\sin^2 x + ... + x^n\sin^n x$.

9. Xây dựng thủ tục để xóa bỏ hết các ký tự trắng dư thừa của một xâu kí tự cho trước (Kí tự trắng đứng vị trí đầu và cuối xâu là dư thừa. Nếu hai kí tự trắng đứng liền nhau thì có một kí tự dư thừa).

10. Xây dựng thủ tục để đổi tất cả các kí tự thành in hoa của một xâu kí tự cho trước.

11. Xây dựng thủ tục để đổi tất cả các kí tự đầu từ thành in hoa, các kí tự khác thành in thường của một xâu kí tự cho trước.

12. Xây dựng hàm đếm số từ có trong một xâu kí tự cho trước.

13. Xây dựng hàm đảo ngược từ cuối cùng thành từ đầu tiên của một xâu kí tự cho trước. (Ví dụ với xâu 'Nguyen Van Anh' truyền vào cho hàm thì giá trị trả về của hàm là 'Anh Nguyen Van').

14. Cho hai xâu dạng số nguyên s1 và s2 (giả sử có độ dài tương đối lớn mà ta không thể dùng thủ tục Val để đổi qua kiểu số được).

15. Xây dựng hàm để tính tổng (theo phép tính số học) của hai xâu dạng số trên. Ví dụ: s1 = '898765' và s2= '888' thì tổng của chúng là '899653'.

16. Xây dựng hàm tính tích của hai xâu s1 và s2.

17. Xây dựng thủ tục TimThay (Var s:string; s1,s2:string) để thực hiện công việc tìm trong xâu s, chổ nào có xâu con s1 thì thay thế bởi xâu s2.

18. Xây dựng thủ tục NhapXauSoNguyen(Var s:String) để nhập xâu biểu diễn số nguyên cho tham biến s. Với yêu cầu: Nếu xâu dạng số s không hợp lệ thì yêu cầu nhập lại. Ví dụ xâu nhập vào là 17650765002 hay -20017 thì chấp nhận, nếu xâu nhập vào là 24e2fg54 hay 1543.005 thì không chấp nhận và yêu cầu nhập lại.

19. Xây dựng thủ tục NhapXauSoNguyen(Var s:String) để nhập xâu biểu diễn số nguyên cho tham biến s. Với yêu cầu: Nhập vào từng kí tự số cho xâu dạng số. Thường xuyên kiểm tra kí tự nhập vào, nếu phát hiện kí tự đó không hợp lệ thì không nhận và cho loa phát lên tiếng "bíp".

20. Xây dựng thủ tục NhapXauSoThuc(Var s:String) để nhập xâu biểu diễn số thực cho tham biến s. Với yêu cầu: Nếu xâu dạng số s không hợp lệ thì yêu cầu nhập lại. Ví dụ xâu nhập vào là 17650765002 hay -20017 thì chấp nhận, nếu xâu nhập vào là 24e2fg54 hay 1543.005 thì không chấp nhận và yêu cầu nhập lại.

CHƯƠNG 7. KIỂU BẢN GHI (RECORD)

Mở đầu:

- Trong các kiểu dữ liệu trước đây, có kiểu dạng đơn trị và cũng có kiểu dạng đa trị. Mỗi đối tượng kiểu đơn trị đều xác định một đại lượng đơn nguyên như các kiểu số, kiểu kí tự, kiểu xâu kí tự, kiểu logíc. Mỗi đối tượng kiểu đa trị chứa nhiều đại lượng và có thể nói các đại lượng đó độc lập với nhau ví dụ như kiểu mảng. Đối với một đối tượng kiểu mảng thì đó là một tập hợp hữu hạn và cố định số lượng phần tử có cùng chung một kiểu. Ở đây chúng ta sẽ đề cập đến một kiểu dữ liệu đa trị mà các đại lượng được lưu trữ trong đó không hẳn phải có cùng chung kiểu dữ liệu, đó là kiểu bản ghi.

- Nội dung chương này giới thiệu kiểu dữ kiểu bản ghi, các thao tác trên kiểu bản ghi cùng các ví dụ minh họa

Mục tiêu: Học xong chương này Sinh viên

- Hiểu được một kiểu dữ liệu đa trị đó là kiểu bản ghi;
- Biết được cách định nghĩa kiểu và khai báo biến.
- Biết được các thao tác trên kiểu bản ghi.
- Hiểu được kiểu bản ghi biến đổi.

7.1. Định nghĩa kiểu và khai báo biến kiểu bản ghi.

7.1.1. Định nghĩa kiểu:

Để định nghĩa kiểu bản ghi ta dùng cú pháp sau:

```
TYPE   TênKiểu = RECORD
                    TênTrường1 : Kiểu1;
                    TênTrường2 : Kiểu2;
                    ......
                    TênTrườngN : KiểuN;
                 END;
```

Trong đó `Tênkiểu` và `TênTrườngi` được đặt theo nguyên tắc đặt tên của Turbo Pascal. `Kiểui` của các trường là một trong các kiểu: số, kí tự, xâu kí tự, đoạn con, mảng hoặc là một kiểu dữ liệu khác đã được định nghĩa trước đó.

Chú ý:

- Nếu một trong các kiểu của trường là kiểu bản ghi thì ta gọi là kiểu bản ghi lồng nhau.

- Nếu có nhiều trường có cùng chung kiểu thì ta có thể kê chung cùng một dòng bằng cách tên các trường đặt cách nhau một dấu phẩy.

Ví dụ:

```
(1) TYPE    HocSinh = RECORD
                         Ten : STR[30];
```

```
                            Toan, Ly, Hoa :  Real;
                          END;
(2) TYPE   ThiSinh = RECORD
                            Ten   :  STR[30];
                            Tuoi  :  Byte;
                            Diem  :  Array[1..3] Of  Real;
                        END;
(3) TYPE   DiaChi =  RECORD
                            SoNha : STR[8];
                            Duong : STR[30];
                        END;
           NhanVien =  RECORD
                            Ten : STR[30];
                            Tuoi : Byte;
                            Dchi : DiaChi;
                          END;
```

7.1.2. Khai báo biến:

Để khai báo biến kiểu bản ghi chúng ta có thể sử dụng một trong hai cách:

Khai báo trực tiếp, ví dụ:

```
VAR  hs1, hs2  :  RECORD
                    Ten:STR[30];
                    Toan, Ly, Hoa:Real;
                  END;
```

Khai báo thông qua kiểu đã được định nghĩa.

```
TYPE   HocSinh = RECORD
                    Ten:STR[30];
                    Toan, Ly, Hoa:Real;
                 END;
VAR   hs1, hs2 : HocSinh;
```

Tuy nhiên khi khai báo kiểu bản ghi cho các tham số hình thức trong phần khai báo các chương trình con thì kiểu đó phải được định nghĩa trước, ví dụ:

```
PROCEDURE   SuaDoi (VAR hs : HocSinh);
FUNCTION    TinhDiem (hs : HocSinh) : Real;
```

Mặc khác nếu khai báo kiểu giá trị trả về của hàm là kiểu bản ghi thì Turbo Pascal không chấp nhận vì đó là kiểu đa trị.

7.2. Các thao tác trên kiểu bản ghi:

7.2.1. Phép gán:

Giả sử có hai biến bản ghi hs1 và hs2 có cùng kiểu dữ liệu. Khi đó phép gán

```
hs2 := hs1;
```

Ý nghĩa: giá trị dữ liệu trong các trường của bản ghi hs1 tương ứng lần lượt sẽ được gán vào cho các trường của bản ghi hs2 (tất nhiên sau khi gán thì giá trị dữ liệu của hai bản ghi hoàn toàn giống nhau).

7.2.2. Truy cập đến dữ liệu của một trường của bản ghi:

Muốn truy cập dữ liệu của một trường nào đó, của một biến bản ghi nào đó thì ta sử dụng cú pháp:

TênBiến.TênTrường

Khi đó chúng ta có thể thao tác với đối tượng này như là một biến có kiểu dữ liệu là kiểu của trường đã được khai báo.

Ví dụ:

Giả sử biến *Ts* có kiểu *ThiSinh* như đã định nghĩa ở trên thì:

Ts. Ten như là một biến có kiểu STR30. Vì vậy khi thao tác với nó ta thao tác như một đối tượng có kiểu STR30, ví dụ:

```
Ts. Ten := 'Nguyen Van Ba' ;
Readln(Ts.Ten);
Writeln(Ts.Ten);
.....
```

Ts. Tuoi như là một biến có kiểu BYTE. Vì vậy khi thao tác với nó ta thao tác như một đối tượng có kiểu BYTE.

Ts. Diem như là một biến có kiểu mảng. Vì vậy khi thao tác với nó ta thao tác như một đối tượng có kiểu mảng đã được định nghĩa kèm theo, ví dụ:

```
Ts. Diem[1] := 5.5 ;
Readln (Ts. Diem[2]);
.....
```

Giả sử biến *Nv* có kiểu *NhanVien* như đã định nghĩa ở trên thì:

Khi cần truy cập đến trường *Ten* và *Tuoi* của biến *Nv* thì ta sử dụng cú pháp như đã trình bày ở trên. Với cách đó thì *Nv. Dchi* như là một biến bản ghi có kiểu *DiaChi*. Vì vậy khi cần truy cập đến trường *SoNha* của trường *Dchi* chứa trong biến *Nv* thì ta dùng cú pháp

Nv. Dchi. SoNha

Đối tượng nầy như là một biến có kiểu STR[8] dùng để truy cập đến trường *SoNha* của trường *Dchi* chứa trong biến *Nv*.

7.2.3. Câu lệnh WITH:

Khi một bản ghi có khá nhiều trường thì việc tuy cập đến một trường phải dùng cú pháp *TênBiến. TênTrường,* như vậy ta phải viết đi viết lại rất nhiều lần

TênBiến. Turbo Pascal cho phép ta giảm thiểu được công việc nầy bằng cách dùng câu lệnh WITH, với cú pháp như sau:

```
WITH  TênBiến  DO
  Begin
     {ở đây chỉ dùng tên trường mà không dùng tên biến}
  End;
```

Ví dụ:

Giả sử biến *Ts* có kiểu *ThiSinh*, cho đoạn chương trình:

```
Readln(Ts. Ten) ;
Readln(Ts. Tuoi) ;
Readln(Ts. Diem[1], Ts. Diem[2], Ts. Diem[3]);
```

Thì đoạn chương trình đó tương đương với đoạn chương trình sau:

```
WITH  Ts  DO
  Begin
     Readln(Ten);
     Readln(Tuoi);
     Readln(Diem[1], Diem[2], Diem[3]);
  End;
```

Giả sử biến *Nv* có kiểu *NhanVien*, cho đoạn chương trình:

```
Readln(Nv.Ten) ;
Readln(Nv.Tuoi) ;
Readln(Nv.Dchi.SoNha);
Readln(Nv.Dchi.Duong);
```

Thì đoạn chương trình đó tương đương với đoạn chương trình sau:

```
WITH  Nv  DO
  Begin
     Readln(Ten);
     Readln(Tuoi);
     WITH  Dchi   DO
       Begin
          Readln(SoNha);
          Readln(Duong);
       End;
  End;
```

Khi đó câu lệnh WITH ở trên được gọi là câu lệnh WITH lồng nhau. Điều này chỉ có khi dùng cho bản ghi lồng nhau.

7.3. Kiểu bản ghi biến đổi.

7.3.1. Định nghĩa kiểu:

Để định nghĩa kiểu bản ghi biến đổi ta dùng cú pháp sau:

```
TYPE  TênKiểu = RECORD
                    Trường1 : Kiểu1;
                    Trường2 : Kiểu2;
                         ......
                    TrườngN : KiểuN;
                    CASE TrườngBiếnThức:KiểuBiếnThức OF
                         GiáTrị1 : (Trường11 : Kiểu11);
                         GiáTrị2 : (Trường22 : Kiểu22);
                         ......
                         GiáTrịM :  (TrườngMM : KiểuMM);
                END;
```

Trong đó *Trường1, Trường2,...., TrườngN* và các kiểu của chúng được gọi là các trường thông thường như đã biết khi định nghĩa kiểu bản ghi, còn *TrườngBiếnThức* được gọi là trường biến thức có kiểu là *KiểuBiếnThức*. Các *GiáTrị1, GiáTrị2*,.., *GiáTrịM* là các hằng giá trị có kiểu là *KiểuBiếnThức*.

Cần chú ý rằng trường biến thức cùng với kiểu của nó được đặt giữa cặp từ khoá CASE và OF, đồng thời trường biến thức bắt buộc phải đặt vào cuối cùng.

7.3.2. Các thao tác trên kiểu bản ghi biến đổi:

Giả sử ta có kiểu bản ghi biến đổi được định nghĩa như sau:

```
TYPE  HinhThe = (vuong, chunhat, tamgiac);
       Chieu = RECORD
                    CASE hinh : HinhThe OF
                    vuong : (canh : Real);
                    chunhat : (dai, rong : Real);
                    tamgiac : (day, cao : Real);
               END;
VAR  doituong : Chieu;
```

Trong đó `HinhThe` là kiểu liệt kê đã được định nghĩa trước và `Chieu` là kiểu bản ghi biến đổi không có trường thường mà chỉ có một trường biến thức có tên là `hinh` với kiểu biến thức là `HinhThe`. Khi đó, vì trường biến thức `hinh` có kiểu `HinhThe` nên nó chỉ có thể lấy các giá trị: `vuong`, `chunhat` hay `tamgiac`. Như vậy, nếu:

`Doituong.hinh := vuong` thì ta chỉ có thể truy cập đến trường `canh` có kiểu dữ liệu Real bằng cách sử dụng kí pháp `Doituong.canh`

`Doituong.hinh := chunhat` thì ta chỉ có thể truy cập đến hai trường `dai, rong` có kiểu dữ liệu Real bằng cách sử dụng kí pháp `Doituong.dai, Doituong.rong`

`Doituong.hinh := tamgiac` thì ta chỉ có thể truy cập đến trường `day, cao` có kiểu dữ liệu Real bằng cách sử dụng kí pháp `Doituong.day, Doituong.cao`

Ví dụ:

```
USES  CRT;
TYPE  HinhThe = (vuong, chunhat, tamgiac);
      Chieu = RECORD
                 CASE hinh    :  HinhThe OF
                      vuong   : (canh : Real);
                      chunhat : (dai, rong : Real);
                      tamgiac : (day, cao : Real);
                 END;
FUNCTION  DienTich (doituong : Chieu) : Real;
   BEGIN
      WITH  doituong  DO
         CASE  hinh  OF
            vuong   : DienTich := canh * canh;
            chunhat : DienTich := dai * rong;
            tamgiac : DienTich := day * cao / 2;
         END;
   END;

PROCEDURE  TinhDienTich;
   VAR  LoaiHinh : Chieu; chon : Byte;
   BEGIN
      Writeln('(1) Tinh dien tich hinh vuong');
      Writeln('(2) Tinh dien tich hinh chu nhat');
      Writeln('(3) Tinh dien tich hinh tam giac');
      Write('Moi ban chon: '); Readln(chon);
      CASE chon OF
        1 : BEGIN
               LoaiHinh.hinh := vuong;
               Write('Cho do dai canh: ');
               Readln(LoaiHinh.canh);
               Write('Dien tich hinh vuong nay la: ');
               Writeln(DienTich(LoaiHinh):6:2);
            END;
        2 : BEGIN
               LoaiHinh.hinh := chunhat;
               Write('Cho biet chieu dai: ');
               Readln(LoaiHinh.dai);
               Write('Cho biet chieu rong: ');
               Readln(LoaiHinh.rong);
```

```
                Write('Dien tich hinh chu nhat nay la: ');
                Writeln(DienTich(LoaiHinh):6:2);
              END;
          3 : BEGIN
                 LoaiHinh.hinh := tamgiac;
                 Write('Cho biet canh day: ');
                 Readln(LoaiHinh.day);
                 Write('Cho biet chieu cao: ');
                 Readln(LoaiHinh.cao);
                 Write('Dien tich hinh tam giac nay la:');
                 Writeln(DienTich(LoaiHinh):6:2);
              END;
       END;
    END;

BEGIN
  Clrscr;
  TinhDienTich;
  Readln;
END.
```

BÀI TẬP CHƯƠNG 7.

1. Mỗi phân số lược lưu trong một bản ghi có hai trường ghi nhận tử số và mẫu số. Hãy tổ chức kiểu dữ liệu tương ứng và viết các thủ tục để cộng, trừ, nhân chia hai phân số và rút gọn phân số.

2. Mỗi số phức được biểu diễn dưới dạng a+ib (trong đó a là phần thực, b là phần ảo. Hãy tổ chức kiểu dữ liệu để lưu số phức và viết các thủ tục để cộng, trừ, nhân chia hai số phức.

3. Mỗi khách hàng của sở điện lực được lưu trong một bản ghi gồm các trường: Họ và tên, địa chỉ, số chữ điện cũ, số chữ điện mới. Hãy tổ chức kiểu dữ liệu tương ứng và viết hàm tính tiền điện tiêu thụ hàng tháng của một khách hàng. Biết rằng số chữ điện tiêu thụ bằng số chữ điện mới trừ số chữ điên cũ, 50 chữ đầu tiên có giá 1000đ mỗi chữ, 30 chữ tiếp theo có giá 1700đ một chữ và số còn lại có giá 3000đ một chữ.

4. Để quản lý điểm học tập của học sinh phổ thông, mỗi môn học người ta cần quản lý các thông tin sau:

Điểm kiểm tra miệng có hệ số 1, gồm 1 cột điểm.

Điểm kiểm tra 15 phút có hệ số 1, gồm 2 cột điểm.

Điểm kiểm tra 1 tiết có hệ số 2, gồm 2 cột điểm.

Điểm thi học kỳ, gồm 1 cột điểm.

Người ta qui định cách tính điểm cuối học kỳ như sau:

Điểm trung bình kiểm tra (TBKT) là điểm trung bình của các điểm kiểm tra.

Điểm trung bình môn (TBM) = (2*TBKT + Điểm thi học kỳ) / 2.

Hãy tổ chức kiểu dữ liệu bản ghi tương ứng và viết chương trình thực hiện các công việc sau:

Nhập danh sách học sinh và các cột điểm tương ứng.

Thông báo lên màn hình danh sách học sinh phải thi lại môn học này (tức là các học sinh có điểm trung bình môn dưới 5).

CHƯƠNG 8. KIỂU TẬP TIN (FILE)

Mở đầu:

- Các kiểu dữ liệu đã được trình bày ở các chương trước đều nhằm mục đích phục vụ cho việc lưu trữ ở bộ nhớ trong (RAM). Các dữ liệu được lưu trữ theo kiểu này mang tính chất tạm thời, phục vụ cho quá trình xử lý thông tin. Trong chương nầy chúng ta được tiếp xúc với một kiểu dữ liệu mới nhằm phục vụ cho việc lưu trữ ngoài (trên đĩa), đó là kiểu dữ liệu tập tin (file). Các dữ liệu được lưu trữ theo kiểu nầy có tính chất vĩnh viễn (giả sử rằng đĩa không bao giờ bị hỏng). Cũng nhờ nó mà ta có thể sao chép dữ liệu từ máy này sang máy khác. Ở đây chủ yếu tìm hiểu hai loại, đó là file định kiểu và file dạng văn bản (Text).

- Nội dung chương này giới thiệu kiểu dữ liệu tập tin cùng các thao tác trên nó.

Mục tiêu: Học xong chương này Sinh viên

- Biết cách định nghĩa và khai báo biến kiểu tập tin.

- Biết cách sử dụng một số thủ tục và hàm chuẩn cho các kiểu tập tin.

- Hiểu được cách thức tổ chức và lưu trữ của một tập tin văn bản và tập tin định kiểu.

8.1. Định nghĩa kiểu và khai báo biến.

8.1.1. Tập tin định kiểu:

Để định nghĩa kiểu ta dùng cú pháp sau:

```
TYPE  TenKieu  =  FILE  OF  KiểuPhầnTử ;
```

Trong đó TenKieu do người sử dụng đặt và KiểuPhầnTử là một trong các kiểu đã biết (Integer, Byte, Word, Real, Char, String, Record) hoặc là kiểu con trỏ sẽ được đề cập đến ở chương sau.

Khi đã định nghĩa kiểu thì ta có thể khai báo biến như sau:

```
VAR    Danh sách các biến : TênKiểu ;
```

Hay có thể khai báo trục tiếp như sau:

```
VAR   Danh sách các biến : FILE OF KiểuPhần Tử ;
```

Trong danh sách các biến thì giữa hai biến được đặt cách nhau một dấu phẩy.

Ví dụ:

```
TYPE   FNGUYEN  =  FILE OF  Integer;
VAR   f, g : FNGUYEN;
```

Hay có thể khai báo trực tiếp:

```
VAR   f, g : FILE OF Integer;
```

Khi đó f và g được gọi là các biến file.

8.1.2. Tập tin dạng văn bản (Text):

Kiểu tập tin dạng văn bản được Turbo Pascal qui định sẵn là TEXT (đồng nghĩa với FILE OF CHAR). Vì vậy ở đây không cần định nghĩa kiểu mà chỉ cần khai báo biến tập tin dạng văn bản như sau:

```
VAR   Danh sách các biến : TEXT;
```

Ví dụ: `VAR` f, g, h : `TEXT` ;

8.1.3. Tập tin không định kiểu:

Khi khai báo các biến có kiểu tập tin không định kiểu thì ta khai báo như sau:

```
VAR     Danh sách biến : FILE;
```

Ví dụ: `VAR` f, g, h : `FILE;`

Khi đó chúng ta chỉ có thể thao tác vật lý trên tập tin được gán tên cho chúng như đổi tên tập tin, xoá tập tin,.... và chỉ có thể truy xuất dữ liệu chứa trong chúng theo từng khối.

8.2. Một số thủ tục và hàm chuẩn.

Các thủ tục và hàm chuẩn được nêu ở đây được dùng chung cho cả tập tin định kiểu và tập tin dạng văn bản.

8.2.1. Gán tên file cho biến file:

Thủ tục `ASSIGN` (BiếnFile, TênFile);

Trong đó:

BiếnFile là một biển kiểu tập tin định kiểu hay tập tin dạng văn bản.

TênFile là một xâu kí tự thể hiện tên của tập tin, có thể có cả đường dẫn.

Ý nghĩa: Thủ tục này nhằm gán tên của một tập tin ở trên đĩa cho biến BiếnFile đang lưu giữ ở bộ nhớ trong.

Ví dụ:

```
ASSIGN (f, 'SOLIEU.DAT');
ASSIGN (g, 'C:\TP\PRO\BAITAP.PAS');
     ..............
WRITE('Cho tên file: '); READLN(St);
ASSIGN (h,  St);
```

8.2.2. Tạo và mở một file mới để ghi dữ liệu:

Thủ tục `REWRITE`(BiếnFile);

Ý nghĩa: Thủ tục này nhằm tạo mới một tập tin trên đĩa với kích thước ban đầu là 0 byte (do dữ liệu chứa bên trong nó là trống), có tên đã gán cho BiếnFile bởi thủ tục `ASSIGN` trước đó và tập tin đó được mở ra để sẵn sàng cho phép ghi dữ liệu vào tập tin.

Nếu TênFile đã gán cho BiếnFile không có đường dẫn thì tập tin sẽ được tạo tại thư mục hiện hành, nếu có đường dẫn thì tập tin sẽ được tạo tại thư mục được chỉ ra theo đường dẫn.

Nếu trên đĩa, tại thư mục nơi tạo ra tập tin mới đã tồn tại tập tin trùng tên thì tập tin cũ bị xoá sạch và thay vào đó là tập tin mới với kích thước ban đầu là 0 byte (tức dữ liệu chứ bên trong nó là trống).

8.2.3. Mở một file đã có trên đĩa:

Thủ tục `RESET`(BiếnFile);

Ý nghĩa: Thủ tục này nhằm mở một tập tin trên đĩa tại thư mục hiện hành hay thư mục theo đường dẫn đã chỉ ra, có tên đã gán cho BiếnFile trước đó để sẵn sàng truy xuất dữ liệu của tập tin.

Nếu tại thư mục mở tập tin không có tập tin có tên đã gán cho BiếnFile tức là gặp lỗi xuất nhập (I/O Error), thì chương trình sẽ thoát ra.

Nếu tập tin được mở là dạng văn bản thì chỉ có thể đọc dữ liệu từ tập tin ra, còn nếu tập tin là dạng định kiểu thì có thể truy xuất dữ liệu bằng cách đọc dữ liệu từ tập tin hay ghi dữ liệu vào tập tin.

8.2.4. Đóng file:

Thủ tục `CLOSE`(BiếnFile);

Ý nghĩa: Thủ tục này nhằm đóng một tập tin đã được mở bởi thủ tục `REWRITE`(BiếnFile) hay `RESET`(BiếnFile) có tên đã gán cho BiếnFile trước đó. Thao tác này hết sức cần thiết khi không làm việc với tập tin nữa để tránh mất dữ liệu của tập tin.

Chú ý: Để tránh việc chương trình sẽ bị thoát ra khi gặp lỗi xuất nhập chúng ta có thể kiểm tra xem tập tin sẽ mở có tồn tại trên đĩa không bởi hàm của người sử dụng được xây dựng như sau:

```
FUNCTION   FileExists(TênFile : STRING) : BOOLEAN ;
     VAR   f : FILE;
     Begin
          { $I- }
          ASSIGN(f, TênFile);
          RESET(f);
          CLOSE(f);
          { $I+ }
          FileExists := (IOResult = 0);
     End;
```

Hàm FileExists sẽ trả về giá trị True nếu tập tin cần mở có trên đĩa, ngược lại cho giá trị False.

Trong hàm trên, chỉ thị dịch { $I- } để tắt việc kiểm tra lỗi xuất nhập dữ liệu, tức là chương trình sẽ không thoát ra khi gặp lỗi I/O Error, sau đó trả lại mặc định là có kiểm tra bởi chỉ thị dịch { $I+ }. Còn hàm chuẩn IOResult sẽ trả về giá trị 0 khi không có lỗi xuất nhập tức là mở tập tin được, ngược lại nhận giá tri khác 0 thể hiện mã lỗi.

8.2.5. Đổi tên một file:

Thủ tục `RENAME`(BiếnFile, TênMới);

Ý nghĩa: Thủ tục này nhằm đổi tên của tập tin đã gán cho BiếnFile bởi thủ tục `ASSIGN` trước đó thành TênMới (TênMới chỉ có tên tập tin mà không có đường dẫn) và thủ thục chỉ được gọi khi tập tin chưa được mở.

Ví dụ: Giả sử tại thư mục hiện hành có tập tin với tên SOLIEU.DAT, để đổi thành tên mới là LUUTRU.DAT ta thực hiện đoạn chương trình sau:

```
VAR   f : FILE;
      ............
ASSIGN(f, ' SOLIEU.DAT ');
RENAME(f, ' LUUTRU.DAT ');
      ............
```

8.2.6. Xoá bỏ một file:

Thủ tục `ERASE`(BiếnFile);

Ý nghĩa: Thủ tục này nhằm xoá ra khỏi đĩa một tập tin có tên đã được gán cho BiếnFile bởi thủ tục `ASSIGN` trước đó, thủ tục này chỉ được thực hiện khi tập tin chưa được mở.

Ví dụ: Giả sử tại thư mục hiện hành có tập tin với tên SOLIEU.DAT, để xoá tập tin nầy khỏi đĩa ta thực hiện đoạn chương trình sau:

```
VAR   f : FILE ;
      ............
ASSIGN(f, 'SOLIEU.DAT');
ERASE(f);
      ............
```

8.3. Tập tin định kiểu

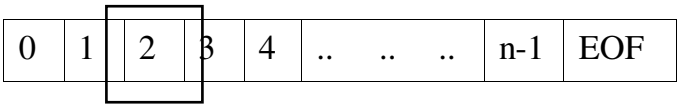
8.3.1. Cách tổ chức lưu trữ của tập tin định kiểu:

Tập tin định kiểu lưu trữ các phần tử có cùng kiểu là KiểuPhầnTử. Các phần tử được lưu trữ kế tiếp nhau và cuối cùng là dấu hiệu kết thúc file (EOF). Số phần tử của tập tin được gọi là độ dài của file. Các phần tử của tập tin lần lược được gán số hiệu theo qui định là 0, 1, 2,.... (như vậy phần tử thứ n của tập tin được gán số hiệu là n-1).

Khi thao tác với tập tin thì chúng ta chỉ có thể ghi dữ liệu vào một phần tử nào đó hoặc đọc dữ liệu tại một phần tử nào đó của tập tin. Ở đây chúng ta đưa thêm

một khái niệm là "cửa số file". Tại một thời điểm cửa sổ file định vị vào một phần tử nào đó của tập tin. Khi đọc dữ liệu từ tập tin hoặc ghi dữ liệu vào tập tin luôn luôn được thực hiện tại phần tử mà cửa sổ file đang định vị.

Minh hoạ:



8.3.2. Các thủ tục và hàm chuẩn:

Khi thao tác với tập tin thì tập tin đó phải được mở (bởi thủ tục `REWRITE` hay `RESET`). Nếu mở bởi thủ tục `REWRITE` thì chỉ có thể ghi dữ liệu vào tập tin, còn nếu mở bởi thủ tục `RESET` thì ban đầu cửa sổ file định vị ở phần tử đầu tiên có số hiệu 0, khi đó ta có thể ghi dữ liệu vào tập tin và đọc dữ liệu từ tập tin ra.

Ngoài các thủ tục và hàm chuẩn được dùng chung cho các kiểu tập tin, sau đây là một số thủ tục và hàm chuẩn chỉ liên quan đến tập tin định kiểu:

Hàm `FILESIZE`(BiếnFile);

Giá trị trả về của hàm là kích thước của tập tin có tên đã gán cho BiếnFile tức là số phần tử thực có trong tập tin. Như vậy các phần tử của tập tin lần lược được đánh số hiệu từ 0 đến `FILESIZE`(BiếnFile)-1.

Hàm `FILEPOS`(BiếnFile);

Giá trị trả về của hàm là số hiệu của phần tử mà cửa sổ file đang định vị của tập tin có tên đã gán cho BiếnFile đang được mở.

Hàm `EOF`(BiếnFile);

Giá trị trả về của hàm là True nếu cửa sổ file của tập tin đang mở có tên đã gán cho BiếnFile đang ở vị trí dấu hiệu kết thúc file. Ngược lại thì giá trị trả về của hàm là False.

Thủ tục `SEEK`(BiếnFile, k);

Thủ tục này nhằm đưa cửa sổ file đến định vị vào phần tử có số hiệu k của tập tin có tên đã gán cho BiếnFile đang được mở.

Thủ tục `READ`(BiếnFile, *Biến*);

Trong đó *Biến* có kiểu là *KiểuPhầnTử* của tập tin. Thủ tục nầy nhằm đọc dữ liệu tại phần tử mà cửa sổ file đang định vị của tập tin có tên đã gán cho BiếnFile đang được mở bởi thủ tục `RESET`, dữ liệu đọc ra được lưu vào Biến. Khi đọc xong thì cửa sổ file tự động định vị vào phần tử tiếp theo của tập tin.

Thủ tục `WRITE`(BiếnFile, BiểuThức);

Trong đó Biểu thức (có thể là hằng, biến, biểu thức hay lời gọi hàm) có kiểu là KiểuPhầnTử của tập tin. Thủ tục này nhằm ghi giá trị có được của BiểuThức vào phần tử mà cửa sổ file đang định vị của tập tin có tên đã gán cho BiếnFile đang

được mở bởi thủ tục RESET hay REWRITE. Khi ghi xong thì cửa sổ file tự động định vị vào phần tử tiếp theo.

8.3.3. Các ví dụ:

Ví dụ 1: Viết chương trình tạo một tập tin mới có tên SOLIEU.DAT tại thư mục hiện hành để thực hiện các công việc sau:

Cho máy lấy ngẫu nhiên một số nguyên dương bé hơn 100, nếu số đó khác 0 thì ghi vào tập tin sau đó lặp lại, nếu số đó bằng 0 thì dừng.

Đọc dãy số đang lưu trong tập tin và ghi lên màn hình.

Sắp xếp các số lưu trong các phần tử của tập tin sao cho khi đọc tuần tự từ tập tin thì ta được dãy số không giảm. Sau đó ghi dãy mới lên màn hình.

Chú ý: không được sử dụng biến kiểu mảng.

Hướng dẫn: Ta cần xây dựng các chương trình con nhằm thực hiện các công việc sau:

Thủ tục TaoFile cho phép lấy ngẫu nhiên các số bé hơn 100 rồi ghi vào tập tin SOLIEU.DAT cho đến số lấy được là số 0 thì kết thúc.

Thủ tục XepThuTu sắp xếp lại dãy theo thứ tự không giảm đang lưu giữ trên tập tin.

Thủ tục DocFile cho hiện dãy số đang lưu giữ trên tập tin ra màn hình.

Chương trình được thể hiện như sau:

```
USES CRT;
     TYPE FNGUYEN = FILE OF WORD;
     VAR f:FNGUYEN;
     PROCEDURE TaoFile(St:STRING);
       VAR f:FNGUYEN; N:WORD;
       Begin
            Assign(f,St);
         Rewrite(f);
         Randomize;
         Repeat
           N:=Random(100);
           If N<>0 Then Write(f,N);
         Until N=0;
         Close(f);
  End;
     PROCEDURE DocFile(St:STRING);
       VAR f:FNGUYEN; N:WORD;
       Begin
         Assign(f,St);
         Reset(f);
         While NOT Eof(f) DO
```

```
            Begin
              Read(f,N); Write(N:4);
            End;
          Close(f);
        End;
      PROCEDURE XepThuTu(St:STRING);
        VAR f:FNGUYEN; N,i,j,a,b:WORD;
        Begin
          Assign(f,St);
          Reset(f);
          N:=Filesize(f);
          For i:=0 To N-2 Do
            Begin
              Seek(f,i);
              Read(f,a);
              For j:=i+1 To N-1 Do
                Begin
                  Seek(f,j);
                  Read(f,b);
                  IF a>b THEN
                    Begin
                      Seek(f,i); Write(f,b);
                      Seek(f,j); Write(f,a);
                      a:=b;
                    End;
                End;
            End;
          Close(f);
        End;
BEGIN
  Clrscr;
  TaoFile('SOLIEU.DAT');
  DocFile('SOLIEU.DAT');
  Writeln;
  XepThuTu('SOLIEU.DAT');
  DocFile('SOLIEU.DAT');
  Readln;
END.
```

Ví dụ 2: Để quản lý kết quả thi tuyển vào lớp 10 của học sinh phổ thông mỗi thí sinh cần lưu giữ các thông tin sau: số báo danh, họ và tên (theo kiểu Việt Nam nhưng không có dấu), điểm thi các môn toán, lý, hoá. Yêu cầu viết chương trình thực hiện các chức năng sau:

Mở một tập tin mới có tên HOCSINH.DAT để nhập họ tên thí sinh và các điểm toán, lý, hoá của thí sinh đó. Việc nhập được lập lại đến chừng nào nhập họ tên thí sinh là xâu rỗng thì kết thúc (chưa đánh số báo danh).

Sắp xếp theo thứ tự alphabet của danh sách có trên tập tin (theo qui cách ưu tiên theo tên sau đó đến họ và chữ lót), sau đó đánh số báo danh thứ tự theo số nguyên 1, 2, 3,.....

Hiển thị danh sách thí sinh lên màn hình kèm theo điểm các môn và tổng điểm.

Hiển thị danh sách thí sinh trúng tuyển lên màn hình kèm theo điểm các môn và tổng điểm. Thí sinh trúng tuyển nếu tổng từ 20 trở lên và không có môn nào có điểm dưới 2.

Hướng dẫn: Ta cần xây dựng các chương trình con nhằm thực hiện các công việc sau:

Thủ tục `TaoFile` cho phép người sử dụng nhập họ tên thí sinh, xâu nhập vào tự động được đổi thành in hoa bởi thủ tục `InHoa`. Tạm thời số báo danh đều được gán bằng 0.

Thủ tục `XepThuTu` sắp xếp lại danh sách theo thứ tự alphabet. Dựa vào hàm `DaoNguoc` để đảo phần tên lên trước, sau đó đến họ và chữ lót nhằm phục vụ cho việc so sánh hai xâu họ tên theo kiểu của Pascal.

Thủ tục `HienDanhSach` cho hiển thị danh sách toàn bộ cùng kết quả điểm.

Thủ tục `DSTrungTuyen` cho hiển thị lên màn hình danh sách trúng tuyển đúng với tiêu chuẩn đã đề ra.

Chương trình được thể hiện như sau:

```
USES CRT;
TYPE  STR30 = String[30];
          HOCSINH = Record
                      SBD:Word;
                      Ten : STR30;
                      T, L, H : Real;
                    End;
          FHOCSINH = FILE OF HOCSINH;
VAR  f:FHOCSINH;
PROCEDURE InHoa(VAR S:STR30);
     VAR i:Word;
     Begin
          For i:=1 To Length(s) Do s[i]:=Upcase(S[i]);
     End;
PROCEDURE TaoFile(St:STRING);
     VAR f:FHOCSINH; Hs:HocSinh; HoTen:STR30; N:Word;
     Begin
```

```
          Assign(f,St);
          Rewrite(f);
          N:=1;
          REPEAT
          Writeln('Nhap ho ten hoc sinh thu',N,':');
          Readln(HoTen);
          InHoa(HoTen);
          If HoTen <> '' Then
               With Hs Do
                    Begin
                         SBD:=0;
                         Ten:=HoTen;
                         Write('Diemtoan: ');
                         Readln(T);
                         Write('Diem ly: ');
                         Readln(L);
                         Write('Diem hoa: ');
                         Readln(H);
                         Write(f,Hs);
                         N:=N+1;
                    End;
          UNTIL HoTen='';
          CLOSE(f);
     End;
FUNCTION  DaoTen(Ten:STR30):STR30;
     VAR s:STR30;
     Begin
          S:='';
          While Ten[Length(Ten)]<>#32 Do
               Begin
                    S:=Ten[Length(Ten)]+S;
                    Delete(Ten,Length(Ten),1);
               End;
          While Ten[Length(Ten)]=#32 Do

Delete(Ten,Length(Ten),1);
          DaoTen:=S+#32+Ten;
     End;
PROCEDURE XepThuTu(St:STRING);
     VAR f:FHOCSINH; N,i,j:WORD; hs1,hs2:HOCSINH;
     Begin
          Assign(f,St);
          Reset(f);
          N:=Filesize(f);
```

```
          For i:=0 To N-2 Do
            Begin
               Seek(f,i); Read(f,hs1);
               For j:=i+1 To N-1 Do
                 Begin
                    Seek(f,j); Read(f,hs2);
                    If   DaoTen(hs1.Ten)>DaoTen(hs2.Ten)
                    Then
                      Begin
                         Seek(f,i);  Write(f,hs2);
                         Seek(f,j);  Write(f,hs1);
                         hs1:=hs2;
                      End;
                 End;
             End;
          N:=1;
          Seek(f,0);
          While Not Eof(f) Do
                Begin
                   Read(f,hs1); hs1.SBD:=N;
                   Seek(f,N-1); Write(f,hs1);
                   N:=N+1;
                End;
          Close(f);
End;
PROCEDURE HienDanhSach(St:STRING);
     VAR f:FHOCSINH; hs:HOCSINH;
     Begin
          Assign(f,St);
          Reset(f);
          While Not Eof(f) Do
               Begin
                    Read(f,hs);
                    With hs Do
                    Writeln(SBD,Ten,T,L,H,T+L+H);
               End;
          CLOSE(f);
     End;
FUNCTION  KiemTra(hs:HOCSINH):Boolean;
  Begin
     With hs Do
     If (T>=2)and(L>=2)and(H>=2)and(T+L+H>=20) Then
     KiemTra:=True  Else KiemTra:=False;
  End;
```

```
PROCEDURE DSTrungTuyen(St:STRING);
  VAR f:FHOCSINH; hs:HOCSINH;
  Begin
     Assign(f,St);
     Reset(f);
     While Not Eof(f) Do
       Begin
          Read(f,hs);
          If KiemTra(hs) Then
         With hs Do
          Begin
           Writeln(SBD,Ten,T:8:2,L:8:2,H:8:2,T+L+H:8:2);
          End;
          CLOSE(f);
       End;
BEGIN
     CLRSCR;
     TaoFile('HOCSINH.DAT');
     XepThuTu('HOCSINH.DAT');
     HienDanhSach('HOCSINH.DAT');
     writeln;
     DSTrungTruyen('HOCSINH.DAT');
     Readln;
END.
```

8.4. Tập tin văn bản

8.4.1. Cách tổ chức lưu trữ:

Như đã nói ở trên, tập tin dạng văn bản chính là tập tin định kiểu với *KiểuPhầnTử* là kiểu CHAR (FILE OF CHAR), có nghĩa mỗi phần tử của tập tin là một kí tự được lưu trữ kế tiếp nhau dưới dạng mã ASCII. Vì vậy ta có thể dùng lệnh TYPE của DOS hay tất cả các phần mềm soạn thảo để đọc dữ liệu của tập tin này.

Khi đọc dữ liệu từ tập tin hay ghi dữ liệu vào tập tin thì dữ liệu đó có thể có kiểu dữ liệu số, kí tự hay xâu kí tự. Trong số các kí tự được ghi vào tập tin có hai kí tự cần được quan tâm đặc biệt đó là kí tự trở về đầu dòng(có mã ASCII 13) và kí tự xuống dòng (có mã ASCII 10), điều này sẽ được trình bày rõ ở các thủ tục chuẩn đọc dữ liệu từ tập tin và ghi dữ liệu vào tập tin.

8.4.2. Các thủ tục và hàm chuẩn:

Ngoài các thủ tục và hàm chuẩn được dùng chung cho các kiểu tập tin, sau đây là một số thủ tục và hàm chuẩn chỉ liên quan đến tập tin dạng văn bản:

Thủ tục `APPEND` (f);

Ý nghĩa: Thủ tục này được dùng để mở tập tin có tên đã được gán cho biến file bởi thủ tục `ASSIGN` trước đó. Khi tập tin được mở thì của sổ file định vị ở vị trí kết thúc của tập tin, từ đó ta chỉ có thể tuần tự ghi thêm dữ liệu vào cuối tập tin.

Thủ tục `READ` (f, Biến1, Biến2,...., BiếnN);

`READLN` (f, Biến1, Biến2,...., BiếnN);

Ý nghĩa: Biến1, Biến2,...., BiếnN (với N≥0) là các biến có kiểu dữ liệu số, kí tự, lôgic hay xâu kí tự. Các thủ tục trên được dùng để đọc dữ liệu từ vị trí của sổ file của tập tin đã được gán tên cho biến file f đang được mở bởi thủ tục `RESET`. Cách đọc phụ thuộc vào kiểu dữ liệu của các biến.

Tuỳ thuộc kiểu của biến mà có những cách đọc khác nhau như sau:

Nếu Biến có kiểu số: Khi đó các số được ghi trên tập tin lần lượt và giữa hai số phải cách nhau ít nhất một kí tự trắng. Mỗi lần đọc xong một số bởi thủ tục `READ` thì cửa sổ file tự động định vị tại vị trí số tiếp theo. Nếu đó là số cuối cùng của dòng thì khi đọc xong cửa sổ file sẽ định vị tại số ở đầu dòng tiếp theo. Mỗi lần đọc xong một số bởi thủ tục `READLN` thì cửa sổ file tự động định vị tại vị trí số ở đầu dòng tiếp theo.

Ví dụ 1: Giả sử tại thư mục hiện hành có tập tin dạng văn bản với tên SOLIEU.DAT chứa dãy các số nguyên được ghi trên hai dòng như sau:

-15 -4 30 -22 101

24 -23 452

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; n:Integer;
BEGIN
     Assign(f,'SOLIEU.DAT');
     Reset(f);
     While Not Eof(f) Do
       Begin
           Read(f,n); Write(n:5);
       End;
     Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

-15 -4 30 -22 101 15 24 -23 452

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; n:Integer;
BEGIN
     Assign(f,'SOLIEU.DAT');
```

```
      Reset(f);
      While Not Eof(f) Do
        Begin
           Readln(f,n); Write(n:5);
        End;
      Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau: -15 15

Chú ý rằng trường hợp đầu dùng thủ tục READ và trường hợp sau dùng thủ tục READLN.

Ví dụ 2: Giả sử tại thư mục hiện hành có tập tin dạng văn bản với tên SOLIEU.DAT chứa dãy các số nguyên được ghi trên hai dòng như sau:

5.0000000000E-01 1.123 1.5000000000E+00

1.12321 2.0000000000E+00 15.5

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; x:Real;
BEGIN
      Assign(f,'SOLIEU.DAT');
      Reset(f);
      While Not Eof(f) Do
        Begin
           Read(f,x); Writeln(x);
        End;
      Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

5.0000000000E-01

1.1230000000E+00

1.5000000000E+00

1.1232100000E+00

2.0000000000E+00

1.5500000000E+01

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; x:Real;
BEGIN
      Assign(f,'SOLIEU.DAT');
      Reset(f);
```

```
      While Not Eof(f) Do
        Begin
           Readln(f,x); Writeln(x:5:3);
        End;
      Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

5.000

1.123

Chú ý rằng trường hợp đầu dùng thủ tục READ và trường hợp sau dùng thủ tục READLN.

Nếu biến có kiểu kí tự: Khi đó nếu đọc bởi thủ tục READ thì kí tự tại của sổ file của tập tin sẽ được đọc và lưu vào Biến, sau khi đọc xong của sổ file tự động chuyển sang kí tự tiếp theo. Các kí tự trở về đầu dòng (#13) và kí tự xuống dòng (#10) vẫn được đọc bình đẳng như các kí tự khác. Nếu đọc bởi thủ tục READLN thì kí tự tại của sổ file của tập tin sẽ được đọc và lưu vào Biến, sau khi đọc xong cửa sổ file tự động chuyển sang kí tự đầu dòng tiếp theo.

Ví dụ : Giả sử tại thư mục hiện hành có tập tin dạng văn bản với tên VANBAN.TXT chứa các dòng văn bản như sau:

aaa bb cc

d ee e

fgh aa

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; ch:Char;
BEGIN
      Assign(f,'VANBAN.TXT');
      Reset(f);
      While Not Eof(f) Do
        Begin
           Read(f,ch); Write(ch);
        End;
      Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

aaa bb cc

d ee e

fgh aa

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; ch:Char;
BEGIN
     Assign(f,'VANBAN.TXT');
     Reset(f);
     While Not Eof(f) Do
       Begin
          Readln(f,ch); Write(ch);
       End;
     Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

adf

Chú ý rằng trường hợp đầu dùng thủ tục READ và trường hợp sau dùng thủ tục READLN.

Nếu biến có kiểu xâu kí tự: Giả sử Biến kiểu xâu kí tự có độ dài tối đa là n (tức có kiểu String[n]). Khi đó nếu đọc bởi thủ tục READ thì bắt đầu tại vị trí của sổ file của tập tin sẽ được đọc một xâu có đúng n kí tự và lưu vào Biến, sau khi đọc xong của sổ file tự động chuyển sang kí tự tiếp theo. Nếu đọc chưa đủ n kí tự thì đã gặp cặp kí tự #13 và #10 thì cũng xem như đọc xong nếu cứ tiếp tục đọc thì chỉ đọc xâu rỗng, không bao giờ cửa sổ file có thể đưa đến sau cặp kí tự #13 và #10 (tức là vị trí đầu tiên của dòng tiếp theo). Nếu đọc bởi thủ tục READLN thì bắt đầu tại vị trí của sổ file của tập tin sẽ được đọc một xâu có đúng n kí tự và lưu vào Biến, nếu đọc chưa đủ n kí tự thì đã gặp cặp kí tự #13 và #10 thì cũng xem như đọc xong. Sau khi đọc xong của sổ file tự động chuyển sang vị trí của kí tự đứng sau cặp kí tự #13 và #10 (tức là đưa đến vị trí đầu tiên của dòng tiếp theo).

Ví dụ : Giả sử tại thư mục hiện hành có tập tin dạng văn bản với tên VANBAN.TXT chứa các dòng văn bản như sau:

aaaaabbbbbcc

dddeefff

gghhhkkkkk

Chú ý rằng ở đây sau mỗi dòng đều có cặp kí tự #13 và #10

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; s:String[5]; I:Integer;
BEGIN
     Assign(f,'VANBAN.TXT');
     Reset(f);
     For I:=1 To 5 Do
```

```
        Begin
           Read(f,s); Writeln('**',s);
        End;
     Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

**aaaaa

**bbbbb

**cc

**

**

Như vậy thể hiện khi đọc 5 lần thì hai lần đầu tiên đọc đủ 5 kí tự cho xâu s, lần thứ ba đọc được 2 kí tự cho xâu s và hai lần cuối chỉ đọc được xâu rỗng (như vậy đọc bởi thủ tục READ không làm cho cửa sổ file vượt qua được cặp kí tự #13 và #10).

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; s:String[5]; I:Integer;
BEGIN
     Assign(f,'VANBAN.TXT');
     Reset(f);
     While Not Eof(f) Do
       Begin
          Readln(f,s); Writeln(s);
       End;
     Close(f);
END.
```

Thì sẽ cho kết quả trên màn hình như sau:

aaaaa

dddee

gghhh

Chú ý rằng nếu thay kiểu String[5] bởi kiểu String[n] với n>=12 hoặc bởi kiểu String thì kết quả trên màn hình sẽ cho như sau:

aaaaabbbbbcc

dddeefff

gghhhkkkkk

Thủ tục WRITE (f, BiểuThức)

WRITELN (f, BiểuThức)

Ý nghĩa: Các thủ tục này được sử dụng để ghi giá trị của BiểuThức (có kiểu dữ liệu số, kí tự, logic hay xâu kí tự) vào tập tin bắt đầu tại vị trí cửa sổ file của tập tin có tên đã được gán cho biến file f bởi thủ tục ASSIGN và đã được mở bởi thủ tục REWRITE hoặc APPEND trước đó. Nếu sử dụng thủ tục WRITE thì khi ghi xong cửa sổ file sẽ định vị định vị ở vị trí kế tiếp, còn nếu sử dụng thủ tục WRITELN thì sau khi ghi xong sẽ tự động ghi thêm cặp kí tự #13 và #10 và cửa sổ file định vị tại vị trí kế tiếp sau đó.

Ví dụ : Giả sử tại thư mục hiện hành có tập tin dạng văn bản với tên VANBAN.TXT chứa các dòng văn bản như sau:

Aaaaa bbb cc

Dddee fff

Chú ý rằng ở đây sau mỗi dòng đều có cặp kí tự #13 và #10

Khi chạy chương trình

```
USES   CRT;
VAR   f :TEXT; s:String[5]; I:Integer;
BEGIN
     Assign(f,'VANBAN.TXT');
     Append(f);
     Write(f,'AAAA BBB');
     Writeln(f,'CC EEEE');
     Writeln(f,'MMMMM NN PPP');
     I:=230; s:='TTTTT';
     Write(I:5, s:8);
     Close(f);
END.
```

Thì nội dung của tập tin sau khi ghi thêm là:

Aaaaa bbb cc

Dddee fff

AAAA BBBCC EEEE

MMMMM NN PPP

((230(((TTTTT

(Chú ý: ở dòng cuối cùng dùng (để thể hiện kí tự trắng)

Hàm EOLN (f)

Ý nghĩa: Giả sử tập tin đã được gán tên cho biến file f và đang mở. Khi đó hàm trả về giá trị True nếu tại thời điểm gọi hàm cửa sổ file đang ở vị trí của cặp kí tự #13 và #10 (kí tự trở về đầu dòng và kết thúc dòng) của dòng văn bản hiện tại. Ngược lại, hàm trả về giá trị False.

Hàm EOF (f)

Ý nghĩa: Giả sử tập tin đã được gán tên cho biến file f và đang mở. Khi đó trả về giá trị True nếu tại thời điểm gọi hàm cửa sổ file đang ở phần tử chứa dấu hiệu kết thúc file. Ngược lại, hàm trả về giá trị False.

BÀI TẬP CHƯƠNG 8.

Tập tin định kiểu

1. Viết chương trình thực hiện các công việc sau, mỗi công việc được viết bởi một chương trình con, sau đó xây dựng menu để chọn lựa các công việc này:

Cho máy lần lượt lấy ngẫu nhiên các số nguyên dương (mỗi số <500) và lưu vào một tập tin định kiểu. Nếu số ngẫu nhiên lấy được là số 0 thì kết thúc.

Bổ sung vào cuối tập tin một số nguyên dương.

Hoán đổi các số lưu trong các phần tử của tập tin sao cho khi đọc dữ liệu từ tập tin ta được dãy số không giảm.

Nếu trong tập tin có hai số giống nhau thì loại bỏ bớt một số sao cho các số còn lại trong tập tin khác nhau từng đôi một.

Hiển thị lên màn hình dãu số có được trong tập tin.

2. Để quản lý điểm học tập của học sinh phổ thông, mỗi môn học người ta cần quản lý các thông tin sau:

Điểm kiểm tra miệng có hệ số 1, gồm 1 cột điểm.

Điểm kiểm tra 15 phút có hệ số 1, gồm 2 cột điểm.

Điểm kiểm tra 1 tiết có hệ số 2, gồm 2 cột điểm.

Điểm thi học kỳ, gồm 1 cột điểm.

Người ta qui định cách tính điểm cuối học kỳ như sau:

Điểm trung bình kiểm tra (TBKT) là điểm trung bình của các điểm kiểm tra.

Điểm trung bình môn (TBM) = (2*TBKT + Điểm thi học kỳ) / 2.

Hãy tổ chức kiểu dữ liệu bản ghi tương ứng và viết chương trình thực hiện các công việc sau, mỗi công việc được viết trong một chương trình con, sau đó tổ chức một menu để chọn lựa các công việc này:

Nhập danh sách học sinh và các cột điểm (khi nào nhập họ tên của học sinh là xâu rỗng thì kết thúc).

Bổ sung thêm họ tên và các cột điểm của một học sinh mới.

Sắp xếp lại các bản ghi trong tập tin theo thứ tự alphabet của họ tên học sinh.

Hiển thị danh sách học sinh cùng các cột điểm và điểm TBM.

Thông báo lên màn hình và in ra giấy danh học sinh phải thi lại môn học này.

3. Để quản lý điểm môn học của học sinh phổ thông như trong bài 2) có một nhược điểm là yêu cầu số cột điểm của các học sinh phải giống nhau, nhưng thực tế số cột

điểm kiểm tra thể khác nhau, ví dụ kiểm tra miệng. Để khắc phục điều này ta không dùng kiểu bản ghi mà tổ chức lưu trữ lại như sau: Mỗi học sinh được lưu trong một phần tử của tập tin định kiểu mà kiểu phần tử là String, theo dạng sau:

Le Van Anh M5P10M7P8T10H9

Với ý nghĩa như sau:

Họ tên của học sinh là: Le Van Anh

Có hai cột điểm miệng là: 5 và 7 (có kí tự M kèm trước)

Có hai cột điểm 15 phút là: 10 và 8 (có kí tự P kèm trước)

Có một cột điểm 1 tiết là: 10 (có kí tự T kèm trước)

Cột điểm thi học kỳ là: 9 (có kí tự H kèm trước)

Hãy viết chương trình thực hiện các công việc sau, mỗi công việc được viết trong một chương trình con, sau đó tổ chức một menu để chọn lựa các công việc này:

Nhập danh sách học sinh.

Bổ sung thêm họ và tên một học sinh mới.

Nhập điểm cho tất cả hoặc một học sinh nào đó.

Kiểm tra xem học sinh nào chưa đủ các loại cột điểm (đủ tất cả các loại cột điểm là phải có các loại điểm kiểm tra miệng, 15 phút, 1 tiết và thi học kỳ). Nếu học sinh nào thiếu thì thông báo lên màn hình và loại cột điểm mà học sinh đó thiếu.

Bổ sung thêm một cột điểm nào đó cho một học sinh nào đó.

Thông báo danh sách lên màn hình cùng điểm TBM đối với các học sinh đầy đủ các loại cột điểm. Nếu học sinh nào thiếu loại cột điểm thì không tổng kết điểm TBM mà chỉ thông báo loại cột điểm còn thiếu.

Tập tin văn bản.

1. Xây dựng thủ tục đọc một tập tin dạng văn bản, với tên tập tin được nhập từ bàn phím, sau đó thông báo lên màn hình số dòng và số từ có trong tập tin đó (có kiểm tra sự tồn tại của tập tin).

2. Viết chương trình cho người sử dụng nhập từ bàn phím một dãy các số nguyên dương, chừng nào số nhập vào là số 0 thì kết thúc. Dãy số nhập vào được lưu trong một tập tin dạng văn bản, giữa hai số đặt cách nhau ít nhất một kí tự trắng. Tên của tập tin được người sử dụng nhập vào từ bàn phím.

3. Giả sử có một tập tin dạng văn bản, lưu dãy các số nguyên dương, trên nhiều dòng và mỗi dòng lưu đúng 10 số, giữa hai số cách nhau ít nhất một kí tự trắng (dòng cuối cùng có thể ít hơn 10 số). Viết chương trình đọc dãy số từ tập tin, tên tập tin được nhập vào từ bàn phím, sau đó thông báo lên màn hình trong tập tin có bao nhiêu số chẵn.

4. Viết chương trình để sao chép một tập tin dạng văn bản đã có trên đĩa thành một tập tin mới. Tên tập tin cũ và mới được nhập từ bàn phím.

5. Viết chương trình đọc một tập tin dạng văn bản đã có trên đĩa. Sau đổi tất cả các kí tự đầu từ thành in hoa, còn các kí tự khác thành in thường.

6. Làm lại bài tập 3) với cấu trúc lưu trữ khác như sau:

Ở đây xâu chứa họ tên và các cột điểm của một học sinh (có kiểu String) sẽ không lưu ở phần tử của một tập tin định kiểu mà lưu trên mỗi dòng của một tập tin dạng văn bản.

CHƯƠNG 9. KIỂU CON TRỎ (POINTER)

Mở đầu :

- Các biến thuộc kiểu dữ liệu đã học như integer, real, mảng, tập hợp, bản ghi… gọi là các biến tĩnh vì chúng được xác định rõ ràng khi khai báo và sau đó được dùng thông qua tên của chúng. Thời gian tồn tại của biến tĩnh cũng là thời gian tồn tại của khối chương trình có chứa khai báo các biến này. Do đó nếu chương trình sử dụng một số lượng lớn các biến tĩnh thì sẽ không đủ bộ nhớ.

- Để tránh lãng phí bộ nhớ Turbo Pascal cho phép dùng biến động (dynamic variable). Các biến này được lưu trữ trong vùng Heap, khi cần chúng có thể được tạo ra để chứa dữ liệu, khi không cần có thể xóa chúng đi để giải phóng bộ nhớ. Biến động không có tên và quản lý biến động này là biến con trỏ .

Mục tiêu: Học xong chương này Sinh viên

- Hiểu được kiểu con trỏ, biết được cách định nghĩa kiểu con trỏ và cách khai báo biến con trỏ.

- Biết cách khởi tạo và loại bỏ một biến động.

- Biết được một số thao tác trên biến con trỏ và biến động.

9.1. Mở đầu.

9.1.1 Phân chia bộ nhớ trong:

Bộ nhớ trong (RAM) được cấu tạo bằng các mạch FLIPFLOP (mỗi mạch được gọi là 1 bít) mô tả được hai trạng thái. Mỗi thanh ghi (REGISTER) được cấu tạo bằng 16 mạch FLIPFLOP nối tiếp nhau, trong đó 8 mạch đầu tạo thành một ô nhớ gọi là Byte thấp và 8 mạch sau tạo thành một ô nhớ gọi là byte cao.

Mỗi ô nhớ được xác định bởi một địa chỉ (adress). Việc xác định địa chỉ cho các ô nhớ được thực hiện như sau:

Tập hợp 65535 ô nhớ tạo thành một SEGMENT, mỗi SEGMENT đều được đánh số gọi là địa chỉ của SEGMENT đó.

Trong mỗi SEGMENT, mỗi ô nhớ đều được đánh số gọi là địa chỉ tương đối hay địa chỉ OFFSET của ô nhớ đó.

Như vậy mỗi ô nhớ được xác định bởi hai địa chỉ, đó là địa chỉ SEGMENT và địa chỉ OFFSET. Hai địa chỉ nàyđược viết dưới dạng cơ số 16 (HEXA) với cú pháp như sau:

<địa chỉ SEGMENT> : <địa chỉ OFFSET>

Ví dụ: Một ô nhớ nằm trong Segment thứ nhất và vị trí củ nó trong segment này là 13, sẽ có địa chỉ là $0001:$000C

Sau đây là các hàm xác định địa chỉ SEGMENT và địa chỉ OFFSET của một Biến, đều có kiểu của giá trị trả về của hàm là WORD:

SEG(<Biến>)

OFS(<Biến>)

Khi chạy một chương trình viết bằng Pascal thì bộ nhớ trong (RAM) được chia thành năm vùng như sau:

Vùng PROGRAM Segment PREFIX, chiếm 256 byte để lưu giữ địa chỉ của các vùng khác, các biến, các hàm, các thủ tục.

Vùng CODE Segment, gồm có hai phần:

MAIN CODE Segment: Vùng chứa đoạn mã của chương trình chính.

UNIT CODE Segment: Vùng chứa đoạn mã của các đơn vị chương trình.

Vùng DATA Segment dùng để lưu các biến chung của chương trình.

Vùng STACK Segment dùng để chứa tạm thời các procedure.

Vùng HEAP dùng để lưu giữ các biến động.

Địa chỉ của ô nhớ đầu tiên của mỗi vùng được gọi là địa chỉ của vùng đó. Các hàm CSEG, DSEG, SSEG có giá trị trả về của hàm kiểu WORD lần lượt là địa chỉ của vùng CODE Segment, DATA Segment, STACK Segment.

9.1.2. Biến con trỏ và biến động:

Biến con trỏ là một biến được đặt tên và được khai báo sau từ khoá VAR, có kiểu con trỏ và được lưu giữ trong vùng nhớ Data Segment. Giá trị được lưu giữ ở biến con trỏ là địa chỉ của một biến động trong vùng HEAP hay một biến nào đó trong DATA Segment.

Biến động là một loại biến không có tên, đó là một vùng nhớ với kích thước cố định được lưu giữ trong vùng HEAP và địa chỉ của nó được lưu giữ ở một biến con trỏ nào đó (còn gọi là được quản lý bởi một biến con trỏ). Khi cần, chúng ta có thể tạo ra một biến động (còn gọi là xin cấp phát một biến động), khi không cần ta có thể xoá đi để trả lại vùng nhớ cho HEAP.

Như vậy, biến động muốn tồn tại thì phải có biến con trỏ quản lý địa chỉ của nó.

Để phân biệt với biến động, các biến có tên được khai báo sau từ khóa VAR sẽ được gọi là các biến tĩnh. Như vậy, biến con trỏ cũng là biến tĩnh.

9.2. Kiểu con trỏ

9.2.1. Định nghĩa kiểu và khai báo biến:

Định nghĩa kiểu:

Kiểu con trỏ được định nghĩa theo cú pháp:

```
TYPE  TênKiểuConTrỏ = <^ KiểuPhầnTử >;
```

Ở đây KiểuPhầnTử là tên của kiểu dữ liệu chứa trong biến động mà biến con trỏ trỏ tới.

Khai báo biến:

Có hai cách khai báo biến con trỏ:

Khai báo với kiểu con trỏ đã được định nghĩa:

```
VAR  p, q : TênKiểuConTrỏ;
```

khai báo trực tiếp:

```
VAR  p, q : <^ KiểuPhầnTử >;
```

Ví dụ:

```
TYPE TRO_NGUYEN = ^ Integer;
VAR p,q : TRO_NGUYEN ;
```

Hay có thể khai báo trực tiếp như sau :

```
VAR  p, q  :  ^Integer ;
```

Khi đó, p và q là các biến con trỏ, quản lý địa chỉ của các biến động chứa dữ liệu là số nguyên (có kích thước là 2 byte).

Ngoài ra, Turbo Pascal cho phép ta khai báo các biến con trỏ không định kiểu với kiểu chuẩn đã được định nghĩa là POINTER. Các biến con trỏ loại này chỉ được sử dụng để quản địa chỉ của tất cả các loại biến động (không phân biệt kiểu của dữ liệu chứa trong biến động đó), ta không thể truy cập dữ liệu tại các biến động thông qua biến con trỏ loại này. Để khai báo biến loại này ta dùng cú pháp như sau:

```
VAR  p, q : POINTER ;
```

9.2.2. Khởi tạo và loại bỏ biến động:

Giả sử p là một biến con trỏ đã được khai báo, khi đó:

Để xin cấp phát một biến động trong vùng HEAP, có địa chỉ được quản lý bởi biến con trỏ p và kích thước của biến động do p qui định, ta dùng:

```
Thủ tục        NEW(p);
```

Để loại bỏ biến động, có địa chỉ đang quản lý bởi biến con trỏ p và đã được tạo ra trước đó bởi thủ tục NEW, ta dùng:

```
Thủ tục        DISPOSE(p);
```

Để lấy kích thước của biến động, ta dùng:

```
Hàm            SIZEOF(p^);
```

Có giá trị trả về của hàm là kích thước của biến động mà biến trỏ con p trỏ tới (đơn vị tính là byte).

Ví dụ: VAR p : ^Integer ; s : ^String ;

Thì giá trị của SIZEOF(p^) sẽ là 2 byte và giá trị của SIZEOF(s^) sẽ là 256 byte.

Để xin cấp phát một biến động trong vùng HEAP, có địa chỉ được quản lý bởi biến con trỏ p và kích thước của biến động là Size (có kiểu Word), ta dùng:

Thủ tục GETMEM(p, Size);

Để loại bỏ biến động có kích thước là Size trong vùng HEAP, có địa chỉ đang quản lý bởi p và đã được tạo ra trước đó bởi thủ tục GETMEM, ta dùng:

Thủ tục: FREEMEM(p, Size);

Chú ý: Các thủ tục DISPOSE và FREEMEM ở trên chỉ có tác dụng giải phóng một biến động đang được quản lý bởi biến trỏ p. Có thể sử dụng một biến con trỏ để tạo ta nhiều biến động, Turbo Pascal cho phép đánh dấu các biến động đó và thực hiện giải phóng cùng một lần. Cụ thể như sau:

Kể từ thời điểm này, các biến động được tạo ra bởi thủ tục NEW(p) hay GETMEM(p,Size)đều được đánh dấu, ta dùng:

Thủ tục MARK (p);

Để giải phóng tất cả các biến động đã được khởi tạo bởi NEW(p)hay GETMEM(p,Size)kể từ khi đã được đánh dấu bởi thủ tục MARK(p), ta dùng:

Thủ tục RELEASE(p) ;

Ví dụ:

```
VAR    p : POINTER;
       p1, p2, p3 : ^Integer ;
..... ;
NEW(p1) ;
MARK(p) ;
NEW(p2) ;
NEW(p3) ;
RELEASE(p) ;
```

{ Kể từ đây, các biến động được khởi tạo sau thủ tục MARK(p) đều được giải phóng. Cụ thể là các biến động đang được quản lý bởi p2 và p3. Còn biến động đang quản lý bởi p1 vẫn tồn tại }

```
..... ;
```

9.2.3. Các thao tác với biến con trỏ và biến động:

Phép lấy địa chỉ của một biến tĩnh: Sử dụng toán tử @, để lấy địa chỉ của biến tĩnh x gán cho biến con trỏ p ta có thể thực hiện như sau:

```
p := @ x ;
```

Phép gán: Ta có thể thực hiện phép gán (giá trị là địa chỉ của biến động) đối với hai biến con trỏ có cùng kiểu với nhau hay một biến con trỏ có kiểu và một biến con trỏ không định kiểu cho nhau. Cụ thể là:

Ví dụ ta có khai báo :

```
VAR  p, q : ^Integer ;
```

Con trỏ p và q trỏ đến biến động chứa dữ liệu kiểu Integer nên có thể gán cho nhau : p := q ;

Khi đó giá trị địa chỉ của biến động đang được quản lý bởi q sẽ được gán cho p. Tức là tại thời điểm này cả p và q đều trỏ đến cùng một biến động.

Con trỏ rỗng: Turbo Pascal qui định một hằng địa chỉ không là NIL, khi con trỏ không có nhiệm vụ quản lý một biến động nào thì ta gán cho nó địa chỉ không:

P := NIL ;

Và khi đó ta nói p là con trỏ rỗng. NIL có thể gán cho bất cứ con trỏ loại nào.

Truy cập dữ liệu tại biến động: Giả sử p là một biến con trỏ, đang trỏ tới một biến động nào đó. Như đã trình bày ở trên là biến động không có tên, ở đây Turbo Pascal cho phép truy cập dữ liệu tại biến động này bởi kí pháp p^. Như vậy ta có thể hiểu p^ như là tên của biến động mà biến con trỏ p đang trỏ tới. Do đó p^ sẽ có kiểu dữ liệu là KiểuPhầnTử của biến con trỏ p và các phép toán thao tác được trên p^ chính là các phép toán có được trên kiểu dữ liệu là KiểuPhầnTử.

Ví dụ:

```
VAR     p : ^Integer ;
        m, n : Integer ;
```

Khi đó p^ sẽ có kiểu dữ liệu là Integer. Như vậy các câu lệnh sau đều hợp lệ:

```
        ..... ;
  New(p) ;
  Readln(p^) ;
```

{nhận một số nguyên nhập từ bàn phím cho biến động đang trỏ bởi p}

```
  Write(p^) ;
```

{viết ra màn hình giá trị của số nguyên đang lưu giữ tại biến động đang trỏ bởi p}

```
        n := 20 ;
        m := 2 * p^ + n ;
             ..... ;
TYPE    HocSinh = RECORD
                    Ten : String[30] ;
                    T, L, H : Real ;
                  END;
VAR   p, q : ^HocSinh ;
      hs : HocSinh ;
```

Khi đó p^, q^ sẽ có kiểu HocSinh (kiểu bản ghi). Như vậy ta có thể thao tác với p^, q^ như là các biến có kiểu bản ghi với bốn trường là: Ten, T, L, H. Do đó các câu lệnh sau đều hợp lệ:

```
.....;
New(p) ;
Write('Nhập họ tên:       '); Readln(p^.Ten);
Write('Nhập điểm toán: '); Readln(p^.T);
Write('Nhập điểm lý:      '); Readln(p^.L);
Write('Nhập điểm hoá:  '); Readln(p^.H);
```

.....;

Hoặc là

.....;

```
New(p) ;
With  p^  Do
  Begin
     Write('Nhập họ tên:        '); Readln(Ten);
     Write('Nhập điểm toán: '); Readln(T);
     Write('Nhập điểm lý:      '); Readln(L);
     Write('Nhập điểm hoá:  '); Readln(H);
  End;
```

.....;

Hoặc là

.....;

```
With   hs   Do
  Begin
     Write('Nhập họ tên:     '); Readln(Ten);
     Write('Nhập điểm toán: '); Readln(T);
     Write('Nhập điểm lý:    '); Readln(L);
     Write('Nhập điểm hoá:  '); Readln(H);
  End;
New(p) ;
p^ := hs ;
```

.....;

Chú ý: Nếu p là con trỏ không định kiểu (p : Pointer) thì p chỉ có nhiệm vụ lưu giữ địa chỉ của biến động nào đó chứ không thể thông qua nó để truy cập dữ liệu tại biến động mà nó trỏ tới được, tức là không thể thao tác với p^.

9.2.4. Một số hàm chuẩn khác:

Hàm `ADDR(Biến);`

Giá trị trả về của hàm là địa chỉ của Biến (giống như sử dụng toán tử @)

Hàm `PTR(seg, ofs : Word);`

Giá trị trả về của hàm là địa chỉ của vùng nhớ có địa chỉ Segment và Offset lần lượt là seg và ofs.

Ví dụ: Mode màn hình được đặt trong vùng nhớ có địa chỉ Segment và Offset là $0000:$0449. Khi đó để lấy giá trị của Mode màn hình ta có thể thực hiện như sau:

```
VAR   p : ^Byte ;
..... ;
p := PTR($0000, $0449);
Writeln(‘ Mode màn hình là: ‘, p^) ;
..... ;
```

Hàm `MEMAVAIL;`

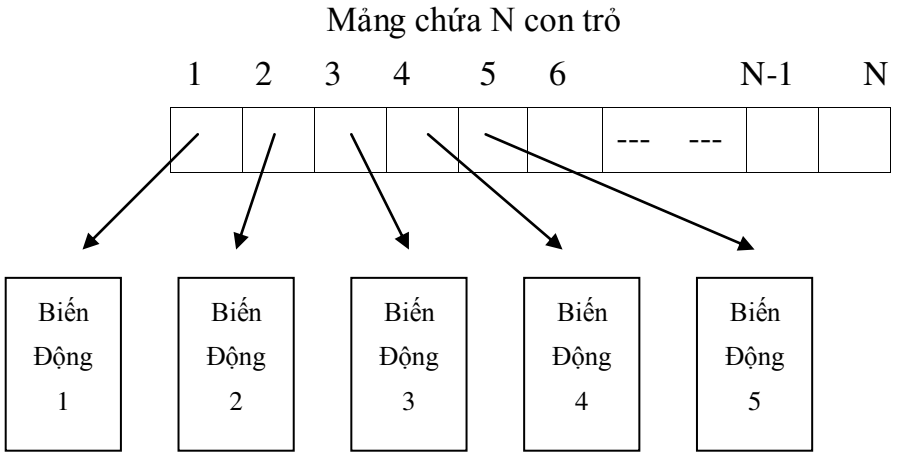
Giá trị trả về của hàm (có kiểu LongInt) là kích thước tối đa của vùng HEAP.

Hàm `MAXAVAIL;`

Giá trị trả về của hàm (có kiểu LongInt) là kích thước còn lại chưa cấp phát của vùng HEAP.

9.2.5. Ví dụ áp dụng:

Giả sử có một kiểu bản ghi với khá nhiều trường, tổng kích thước của một biến có kiểu bản ghi này là K (bytes). Nếu ta tổ chức mảng có N phần tử để lưu các bản ghi trên thì kích thước của mảng phải là N*K (bytes). Như vậy nếu sử dụng không hết các phần tử của mảng thì sẽ lãng phí bộ nhớ rất nhiều khi K khá lớn. Ngược lại, nếu chọn N bé hơn thì trong quá trình xử lý có thể không còn chổ để lưu thêm bản ghi mới. Vì vậy ta tổ chức một mảng gồm N phần tử để lưu các con trỏ (mỗi phần tử chỉ chiếm 2 bytes) trỏ đến các biến động có kiểu bản ghi ở trên. Như vậy khi nào cần lưu trữ thì xin cấp phát biến động để lưu trữ bản ghi, khi nào không cần thì giải phóng biến động đó. Rõ ràng bằng cách này chúng ta tiết kiệm được bộ nhớ.



Với hình minh hoạ như trên cho thấy ta tổ chức được một danh sách các biến động được quản lý bởi mảng các con trỏ.

Một tính toán nhỏ cho ta thấy sự tiết kiệm được bộ nhớ như đã nêu ở trên:

Giả sử N=100, mỗi bản ghi có kích thước 20 bytes.

Nếu tổ chức mảng chứa các bản ghi thì phải mất: 100*20=2000 bytes. Nhưng thực tế chỉ có 5 bản ghi, chiếm 5*20=100 bytes. Như vậy số byte dành chỗ nhưng không sử dụng là: 2000-100=1900 bytes.

Nếu tổ chức mảng con trỏ để quản lý danh sách thì tại thời điểm nầy số lượng bytes đã chiếm dụng là: 100*2 + 5*20 = 300 bytes, thực sự số bytes lãng phí là: 95*2=190 bytes (do 95 phần tử chứa con trỏ của mảng chưa sử dụng).

Ta xét ví dụ cụ thể sau:

Viết chương trình cho người sử dụng nhập danh sách thí sinh thi tuyển vào lớp 10, gồm họ và tên, điểm các môn thi toán, lý, anh văn. Số lượng thí sinh do người sử dụng nhập từ bàn phím (số thí sinh bé hơn hoặc bằng 1000). Danh sách thí sinh lưu vào các biến động được quản lý bởi mảng các con trỏ. Yêu cầu sắp xếp danh sách theo thứ tự giảm dần của tổng điểm và hiển thị danh sách đó lên màn hình.

```
USES crt;
TYPE THISINH = Record
                    Ten:String[30];
                    T,L,AV:Real;
               End;
MANG_CONTRO=Array[1..1000] Of ^THISINH;
VAR  p:MANG_CONTRO;
          Max:Word;

PROCEDURE Wait;
     Begin
          Writeln;
          Write('An phim bat ky de tiep tuc');
          Repeat Until Keypressed;
     End;

PROCEDURE HienThiDS;
Var i:Word;
     Begin
          For i:=1 To Max Do writeln(i,'. ',p[i]^.Ten);
          Writeln;
     End;

PROCEDURE NhapDS(Var Max:Word; Var p:MANG_CONTRO);
  Var i:Word;
  Begin
     Write('Nhap so thi sinh: '); Readln(Max);
     For i:=1 To Max Do
       Begin
```

```
            New(p[i]);
            With p[i]^ Do
              Begin
                 Write(i,'. Ho va ten: '); Readln(Ten);
                 Write('Diem Toan:     '); Readln(T);
                 Write('Diem Ly:       '); Readln(L);
                 Write('Diem Anh van:  '); Readln(AV);
                 Writeln;
              End;
         End;
       Writeln('Da nhap xong danh sach');
  End;

PROCEDURE XepThuTu(Max:Word; Var p:MANG_CONTRO);
  Var i,j:Word; q:POINTER; A,B:Real;
  Begin
     Write('Dang xep thu tu giam dan theo tong diem');
     For i:=1 To Max-1 Do
       For j:=i+1 To Max Do
          Begin
            A:= p[i]^.T+p[i]^.L+p[i]^.AV;
            B := p[j]^.T+p[j]^.L+p[j]^.AV ;
            If A < B Then
               Begin
                 q:=p[i]; p[i]:=p[j]; p[j]:=q;
                 Writeln('Da xep thu tu xong');
               End;
            End;
  End;

BEGIN
     NhapDS(Max,p);
     HienThiDS;
     Wait;
     XepThuTu(Max,p);
     HienThiDS;
     Wait;
END.
```

9.3. Danh sách liên kết.

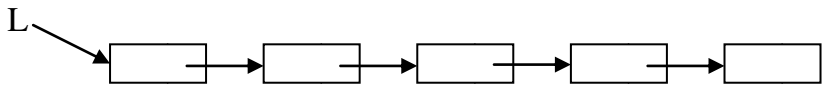
9.3.1. Khái niệm về danh sách liên kết:

Danh sách liên kết là một tập hợp có thứ tự các biến động. Mỗi biến động có kiểu dữ liệu bản ghi, có tối thiểu hai trường, một trường hoặc nhiều trường dùng để lưu dữ liệu và có một trường (dạng con trỏ) để lưu địa chỉ của biến động sau nó (trỏ

đến biến động sau nó). Biến động đầu tiên được quản lý bởi một con trỏ L và con trỏ chứa trong biến động sau cùng là con trỏ rỗng. Khi đó, ta gọi đây là danh sách liên kết L.

Qui ước thêm: ta gọi danh sách là danh sách rỗng nếu danh sách đó không có biến động nào. Nếu L là danh sách rỗng thì L lưu địa chỉ rỗng, tức là L=NIL.

Minh hoạ như sau:



Như vậy khi sử dụng danh sách liên kết ta vẫn tiết kiệm được bộ nhớ nhiều hơn so với cách tổ chức mảng chứa con trỏ để quản lý danh sách như đã trình bày ở trên. Cụ thể là tiết kiệm thêm được vùng nhớ trong mảng chứa các con trỏ chưa được sử dụng.

9.3.2. Cách định nghĩa kiểu cho các nút (biến động):

Trong Turbo Pascal người ta khai báo kiểu dữ liệu cho các nút theo cú pháp như sau:

```
TYPE  KiểuConTrỏ : < ^KiểuBảnGhi > ;
      KiểuBảnGhi  =  RECORD
                         TênTrường1 : <Kiểu1> ;
                         TênTrường2 : <Kiểu2> ;
                         ...................
                         TênTrườngN : <KiểuN> ;
                         ConTrỏ  : KiểuConTrỏ ;
                       END ;
```

Trong đó, KiểuConTrỏ để khai báo các biến con trỏ quản lý địa chỉ của các nút. KiểuBảnGhi để khai báo các nút (biến động). Để minh họa, ta đưa ra các ví dụ sau:

Ví dụ 1: Để xây dựng danh sách liên kết chứa các nút, mỗi nút chứa một số nguyên dương, ta định nghĩa các kiểu sau:

```
TYPE      TRO = ^NUT;
               NUT = Record
               So : Word;
               Tiep : TRO;
           End;
```

Ví dụ 2: Để xây dựng danh sách liên kết chứa các nút, mỗi nút chứa họ và tên, điểm các môn thi toán, lý, anh văn của một thí sinh, ta định nghĩa các kiểu sau:

```
TYPE      DSTHISINH = ^THISINH;
               THISINH = Record
                           Ten : String[30];
```

```
                              T,L,AV:Real;
                              Tiep : DSTHISINH;
                            End;
```

Hay là

```
TYPE      DULIEU = Record
                     Ten : String[30];
                     T,L,AV:Real;
                   End;
          DSTHISINH = ^THISINH;
          THISINH = Record
                      DL : DULIEU;
                      Tiep : DSTHISINH;
                    End;
```

9.3.3. Các Ví dụ:

Viết chương trình cho máy lấy ngẫu nhiên một dãy gồm N số nguyên dương (N được nhập từ bàn phím và mỗi số của dãy < 500). Dãy số được lưu trong các nút của một danh sách liên kết. Trả lời trên màn hình, dãy số có bao nhiêu số chẵn.

```
USES     CRT;
    TYPE      TRO = ^NUT;
                   NUT = Record
                             So : Word;
                             Tiep : TRO;
                           End;
    VAR  L : TRO;
    PROCEDURE  TaoDanhSach(Var L:TRO);
         Var  p,M:TRO; N,i:Word;
         Begin
              Write('Nhap N= '); Readln(N);
              Randomize;
              L := Nil;
              For i:=1 To N Do
                   Begin
                        New(p);
                        p^.so := Random(500);
                        p^.Tiep := Nil;
                        If L=Nil Then L:=p
                        Else M^.Tiep := p;
                        M := p;
                   End;
         End;
    FUNCTION  SoChan(L:TRO):Word;
         Var  p:TRO; dem:Word;
         Begin
```

```
          Dem := 0;
          P := L;
          While p<>Nil Do
               Begin
                    If p^.So mod 2 <> 0 Then dem:=dem+1;
                    P:=p^.Tiep;
               End;
          Sochan:=dem;
     End;

BEGIN
     TaoDanhSach(L);
     Write('So luong so chan la: ',SoChan(L));
     Readln
END.
```

Viết chương trình cho người sử dụng nhập danh sách thí sinh thi tuyển vào lớp 10, gồm họ và tên, điểm các môn thi toán, lý, anh văn, đến chừng nào nhập họ tên thí sinh là xâu rỗng thì kết thúc. Danh sách thí sinh lưu vào các biến động của một danh sách liên kết quản lý bởi biến trỏ L. Sau đó cho hiển thị danh sách thí sinh lên màn hình. Từ danh sách đã có hãy hoán đổi các bản ghi trong các biến động sao cho khi duyệt danh sách ta được tổng điểm của thí sinh được xếp theo thứ tự giảm dần của tổng điểm.

```
USES CRT;
TYPE      DSTHISINH = ^THISINH;
          THISINH = Record
                         Ten : String[30];
                         T,L,AV:Real;
                         Tiep : DSTHISINH;
                    End;

VAR  L:DSTHISINH;
          Max:Word;

PROCEDURE Wait;
     Begin
          Writeln;
          Write('An phim bat ky de tiep tuc');
          Repeat Until Keypressed;
     End;

PROCEDURE HienThiDS(L:DSTHISINH);
     Var p:DSTHISINH;
     Begin
```

```
          P:=L;
          While p<>NIL DO
               Begin
                    Writeln(i,'. ',p^.Ten);
                    P:=P^.Tiep;
               End;
          Writeln;
     End;

PROCEDURE NhapDS(Var L:DSTHISINH);
  Var M,p:DSTHISINH; St:String;
  Begin
     L:=NIL;
     Write('Nhap so thi sinh: ');
       Repeat
          Write('Ho va ten: '); Readln(st);
          If st<>" Then
            Begin
               New(p);
               With p^ Do
                 Begin
                    Ten:=st;
                    Write('Diem Toan:'); Readln(T);
                    Write('Diem Ly:  '); Readln(L);
                    Write('Diem Anh van:'); Readln(AV);
                    If L=NIL Then L:=p Else M:=p;
                    M:=p;
                 End;
            End;
          Until st=";
          Writeln('Da nhap xong danh sach');
     End;

PROCEDURE XepThuTu(Max:Word; Var p:MANG_CONTRO);
  Var i,j:Word; q:POINTER; A,B:Real;
  Begin
     Write('Dang xep thu tu giam dan theo tong diem');
       For i:=1 To Max-1 Do
          For j:=i+1 To Max Do
            Begin
               A:= p[i]^.T + p[i]^.L + p[i]^.AV;
               B:= p[j]^.T + p[j]^.L + p[j]^.AV;
                 If A < B Then
                    Begin
```

```
                        q:=p[i]; p[i]:=p[j]; p[j]:=q;
                    End;
                End;
            Writeln('Da xep thu tu xong');
    End;

BEGIN
     NhapDS(Max,p);
     HienThiDS;
     Wait;
END.
```

BÀI TẬP CHƯƠNG 9

1. Giả sử trên đĩa, tại thư mục hiện thời có tập tin dạng văn bản với tên DAYSO.DAT chứa dãy số nguyên, giữa hai số cách nhau ít nhất một kí tự trắng. Viết chương trình thực hiện các công việc sau:

Đọc dãy số và lưu vào một danh sách liên kết, mỗi nút chứa một số nguyên.

Trên danh sách nếu có hai nút chứa số nguyên bằng nhau thì loại bỏ bớt một nút cho đến khi trên danh sách chỉ còn các số nguyên khác nhau từng đôi một.

Trên cơ sở dãy số đã có được sau khi loại bỏ như câu b), hãy hoán đổi số lưu trong các nút để danh sách lưu dãy số theo thứ tự tăng dần.

Cho nhập từ bàn phím nột số nguyên N, nếu N chưa được lưu ở nút nào thì tìm vị trí để chèn vào một nút mới chứa N sao cho dãy số lưu trong danh sách vẫn bảo đảm theo thứ tự tăng dần.

Hiển thị dãy số có trong danh sách lên màn hình.

2. Để xử lý số nguyên lớn (ví dụ 536523472378239) người ta tổ chức cấu trúc dữ liệu như sau: Mỗi số nguyên được lưu trong một danh sách liên kết, mỗi nút lưu một chữ số của số nguyên đó, được lưu theo chiều ngược của số nguyên (tức chữ số hàng đơn vị lưu ở nút đầu tiên, các nút tiếp theo là chữ số hàng chục, hàng trăm, hàng ngàn...) Hãy viết chương trình thực hiện các công việc sau:

Nhập từ bàn phím một số nguyên lớn.

Hiển thị một số nguyên lớn lên màn hình.

Cộng, trừ, nhân, chia hai số nguyên lớn.

Mỗi công việc được viết trong một chương trình con và xây dựng menu để người sử dụng chọn công việc.

TÀI LIỆU THAM KHẢO

[1] Hồ Sĩ Đàm, Trần Văn Đạo, ***Tin học 11***, NXB Giáo dục, 2007

[2] Quách Tuấn Ngọc, ***Ngôn ngữ lập tnh Pascal -*** *Tập 1 & 2*, Năm 1994

[3] Đỗ Ngọc Phương, ***Giáo trình Pascal - Tập 1 & 2***, Trường Tin học & Ngoại ngữ TP Hồ Chí Minh.