PGS. PTS. BÙI THẾ TÂM

GIÁO TRÌNH

TURBO PASCAL 7.0

LÝ THUYẾT, BÀI TẬP VÀ LỜI GIẢI

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI
HÀ NỘI 2004

T.S. BÙI THẾ TÂM

GIÁO TRÌNH TURBO PASCAL 7.0

- Lập trình Pascal cơ bản
- Kỹ thuật lập trình Pascal nâng cao
- Cấu trúc dữ liệu và giải thuật thể hiện trên Pascal
- Bài tập và lời giải mẫu

Dùng cho :

- Sinh viên các ngành khoa học tự nhiên và kỹ thuật
- Học sinh phổ thông chuyên tin

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI
HÀ NỘI

Lời mở đầu

Turbo Pascal là một ngôn ngữ lập trình có cấu trúc, nó được dùng phổ biến ở nước ta hiện nay trong công tác giảng dạy, lập trình tính toán, đồ hoạ. Turbo Pascal được dùng trong chương trình giảng dạy Tin học ở hầu hết các trường đại học, cao đẳng, trung học phổ thông.

Cuốn sách này là các bài giảng của Tác giả về ngôn ngữ lập trình Pascal (cơ bản và nâng cao) cho sinh viên một số trường đại học và các lớp chuyên tin phổ thông. Giáo trình chia thành ba phần:

- Phần Pascal cơ bản (chương 1-4): trình bày các khái niệm cơ bản, lệnh rẽ nhánh và lặp, dữ liệu kiểu mảng, kiểu xâu ký tự và kiểu tập hợp, hàm và thủ tục.

- Phần Pascal nâng cao (chương 5-7, 9-11): dữ liệu kiểu bản ghi và kiểu tệp, unit Crt, unit Graph, unit DOS, unit tự tạo.

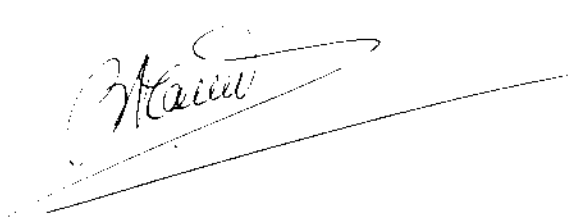
- Phần cấu trúc dữ liệu và giải thuật thể hiện trên Pascal (chương 8, 12-13): con trỏ, danh sách liên kết đơn và kép, ngăn xếp, hàng đợi, cây tìm kiếm nhị phân, các bài toán tổ hợp, các bài toán trên đồ thị.

Cuốn sách này là sự kết hợp, có chỉnh lý và bổ sung của hai cuốn sách đã có nhiều năm nay của tác giả: ***Giáo trình Turbo Pasacl 7.0***, ***Bài tập lập trình Turbo Pascal 7.0***. Cuối mỗi chương đều có các bài tập chọn lọc để luyện tập phần lý thuyết, hầu hết các bài tập đều có lời giải cho ở cuối sách.

Đối tượng của cuốn sách là các bạn sinh viên, học sinh phổ thông, học sinh các lớp chuyên tin, học viên các lớp cao học thuộc các ngành khoa học kỹ thuật đang học các môn : ngôn ngữ lập trình Pascal cơ bản và nâng cao, cấu trúc dữ liệu và giải thuật. Cuốn sách cũng là tài liệu tham khảo cho giáo viên dạy các môn này, cho các bạn tự học không có điều kiện tới lớp.

Tác giả rất mong nhận được sự đóng góp ý kiến của các thầy giáo, sinh viên và các chuyên gia về Tin học để cho cuốn sách được hoàn thiện hơn. Ý kiến đóng góp xin gửi về: Bùi Thế Tâm, Viện Toán học, 18 Hoàng Quốc Việt, 10307 Hà nội, E-mail bttam@math.ac.vn.

Tác giả



T.S. Bùi Thế Tâm

Chương 1

Các khái niệm cơ bản

Pascal là một loại ngôn ngữ lập trình bậc cao, hiện đang được dùng phổ biến ở nước ta trong công tác giảng dạy, lập trình tính toán, xử lý văn bản, đồ hoạ và âm thanh. Ngôn ngữ Pascal do Niklaus Wirth, giáo sư điện toán trường đại học kỹ thuật Zurich (Thuỵ sĩ), đề xuất vào năm 1970 với tên gọi Pascal để kỷ niệm nhà toán học và triết học nổi tiếng người Pháp Blaise Pascal.

Ngôn ngữ Pascal có nhiều ưu điểm: ngữ pháp, ngữ nghĩa đơn giản rõ ràng; cấu trúc chương trình chặt chẽ, dễ hiểu; chương trình dễ sửa, cải tiến. Lúc đầu Pascal được sáng tạo ra để giảng dạy lập trình. Trong quá trình phát triển Pascal đã tỏ ra hơn hẳn nhiều ngôn ngữ bậc cao khác và đến nay nó đã trở thành một ngôn ngữ mạnh được dùng rộng rãi trong rất nhiều lĩnh vực khác nhau. Dựa trên Pascal chuẩn, nhiều tổ chức và công ty máy tính đã phát triển và tạo ra các sản phẩm phần mềm Pascal với nhiều phần thêm bớt khác nhau, trong đó hệ Turbo Pascal (TP) của hãng Borland tỏ ra có nhiều ưu điểm hơn cả và hiện nay nó đã trở thành phổ biến nhất trên thế giới. TP đã được cải tiến qua nhiều phiên bản 1.0, 2.0, 3.3, 4.0, 5.5 (1989), 6.0 (1990), 7.0 (1992).

Turbo Pascal 7.0 chứa trên hai đĩa mềm 1.44 MB. Tuy nhiên nếu chỉ sử dụng chức năng tính toán thì chỉ cần hai tệp chính: Turbo.exe và Turbo.tpl. Muốn dùng đồ hoạ cần thêm các tệp: Graph.tpu, EgaVga.bgi, các tệp phông chữ `*.chr` (Bold.chr, Goth.chr, Litt.chr, Sans.chr, Trip.chr . . .). Tất cả các tệp này chiếm khoảng 600 KB.

1. Khởi động Turbo Pascal và cách chạy một chương trình

Giả sử các tệp của TP lưu trong thư mục `C:\TP7` và ta đang ở Windows 98. Để khởi động TP ta dùng lệnh Start / Program / MS-DOS Prompt, xuất hiện dấu nhắc DOS, dùng lệnh `C:\TP7\BIN>TURBO`, xuất hiện màn hình làm việc của TP.

Phía trong của màn hình làm việc là cửa sổ soạn thảo với tên tệp mặc định là Noname00.pas, trên đỉnh màn hình là menu chính với các mục:

```
File Edit Search Run Compile Debug Tools Options Window Help
```

Dưới đáy màn hình ghi các phím chức năng thường dùng nhất khi làm việc với TP. Để chọn một mục trong menu chính ta ấn Alt + chữ cái đầu tiên của mục, xuất hiện menu dọc ứng với mục vừa chọn. Để chọn một chức năng của menu dọc ta dùng các phím mũi tên để di chuyển hộp sáng và ấn Enter, nếu không muốn chọn mục nào thì ấn ESC để thoát khỏi menu dọc. Nhiều mục của menu dọc có các phím gõ tắt tương ứng, khi đó ta có thể gõ các phím này mà không cần vào hệ thống menu. Ví dụ khi đang soạn thảo chương trình để ghi tệp vào đĩa ta chỉ cần ấn F2 thay cho việc vào hệ thống menu : File / Save.

Các lệnh thường dùng khi làm việc với Turbo Pascal 7.0

- **File/ New**: soạn một tệp mới, tên tệp mặc định là NONAMExx.PAS (xx thay cho một số từ 00 đến 99). Khi ghi tệp Noname lên đĩa máy sẽ nhắc đặt tên cho tệp này.
- **File/ Open (hay ấn F3)**: mở tệp đã có trên đĩa để làm việc.
- **File/ Save (hay ấn F2)**: ghi tệp đang soạn thảo lên đĩa.

- **File/ Save as** : ghi tệp lên đĩa với tên khác, vào tên tệp mới.

- **Run/ Run (hay ấn Ctrl+F9)**: thực hiện việc dịch, liên kết và chạy tệp chương trình đang soạn thảo trong cửa sổ hoạt động. Kết quả biên dịch nằm trong bộ nhớ hay trên đĩa tuỳ theo mục chọn Destination trong menu Compile quyết định.

- **Alt+F5**: xem lại màn hình kết quả chạy chương trình, ấn phím bất kỳ để trở về màn hình làm việc của TP.

- **Edit/ Copy (hay ấn Ctrl+Ins)**: chép khối văn bản vào vùng đệm. Cách đánh dấu khối: đưa con trỏ vào đầu khối, ấn phím Shift và dùng các phím mũi tên để mở rộng vệt sáng đánh dấu khối; hoặc ấn Ctrl+K B ở đầu khối và Ctrl+K K ở cuối khối.

- **Edit/ Cut (hay ấn Shift+Del)**: xoá khối và chép nó vào vùng đệm.

- **Edit/ Paste (hay ấn Shift+Ins)**: chép khối từ vùng đệm vào chỗ con trỏ.

- **Ctrl + Y**: xoá dòng có con trỏ.

- **Search/ Replace (hay ấn Ctrl+Q A)**: tìm một dãy ký tự và thay thế bằng dãy ký tự khác.

- **Compile/ Compile (ấn Alt+F9)**: chỉ tiến hành dịch chương trình mà không chạy. Nếu khi dịch Destination là Memory thì tệp kết quả dịch chỉ lưu trong bộ nhớ, nếu Destination là Disk thì tệp kết quả dịch lưu trên đĩa với phần mở rộng là EXE. Để thay đổi hướng kết quả dịch dùng lệnh Compile/ Destination.

- **Alt + số hiệu cửa sổ**: chuyển về cửa sổ có số hiệu đã ấn. **F5**: phóng to cửa sổ hiện hành ra toàn màn hình. **Alt+F3**: đóng cửa sổ hiện hành.

- **File/ DOS Shell** : tạm thời thoát khỏi môi trường TP về dấu nhắc của DOS để thực hiện các lệnh của DOS (ví dụ chạy một tệp chương trình có đuôi EXE). Trở lại môi trường TP: gõ EXIT tại dấu nhắc của DOS.

- **File/ Exit (hay ấn Alt+X)**: thoát khỏi TP.

Các bước để chạy một chương trình

Trong môi trường TP dùng lệnh File/ New mở một tệp mới, soạn thảo văn bản chương trình, ví dụ chương trình **InDong.pas** in một dòng chữ lên màn hình:

```
Begin
Writeln('Ha Noi la thu do cua Cong hoa Xa hoi Chu nghia Viet Nam');
End.
```

Soạn xong ghi tệp vào đĩa: ấn F2, vào tên tệp nếu ghi lần đầu. Dịch chương trình xem có lỗi không: Alt+F9, nếu có lỗi thì sửa (con trỏ dừng lại ở chỗ có lỗi, dòng màu đỏ trên đỉnh màn hình thông báo nguyên nhân lỗi) và dịch lại. Chạy chương trình: Ctrl+F9.

2. Cú pháp và ngữ nghĩa

Một ngôn ngữ bất kỳ, từ ngôn ngữ của con người đến ngôn ngữ của máy tính, đều được xây dựng dựa trên một bộ các ký tự. Các ký tự ghép lại thành các từ, tập hợp các từ ghép theo những quy tắc cú pháp nhất định tạo thành những câu để diễn tả một nội dung nào đó, nội dung ấy là ngữ nghĩa của câu.

Bộ ký tự. Ngôn ngữ Pascal sử dụng các ký tự sau: 52 chữ cái in hoa và in thường, (A..Z, a..z), 10 chữ số (0..9) và các ký tự + - * / = < > ' " ( ) [ ] { } . , : ; ? ! @ # $ % ^ & và dấu nối chân (_).

Từ khoá (key word) là những từ dành riêng được TP dùng với ý nghĩa nhất định. Các từ khoá thường dùng: `and, array, begin, case, const, div, do, downto, else, end, file, for, forward, function, goto, if, in, label, mod, not, of, or, procedure, program, record, repeat, set, string, then, to, type, unit, until, uses, var, while, with`. Từ khoá cần được viết đúng (chữ thường hay chữ hoa đều được) và viết tách khỏi các ký tự khác.

Tên (indentifier) là một dãy ký tự được tạo thành từ các chữ cái, chữ số và dấu nối chân (_) dùng để đặt tên cho các đại lượng trong chương trình như tên biến, tên hằng, tên kiểu dữ liệu, tên mảng, tên hàm, tên bản ghi, tên chương trình, tên con trỏ... Ký tự đầu tiên của tên không được là chữ số. Tên không được có dấu cách và không được trùng với từ khoá. Tên có thể có độ dài tuỳ ý nhưng chỉ có 63 ký tự đầu tiên là có nghĩa. Ví dụ tên đúng: `Alpha, x1, _z, pt_bac_2`. Tên sai: `pt bac2, 4abc, f(x), in, x+y`.

Tên chuẩn là tên đã được TP định nghĩa trước dùng để chỉ các hàm, hằng, biến, thủ tục, thư viện của TP như `Boolean, Byte, Char, Integer, Real, False, True, Text, Abs, Arctan, Chr, Cos, Sin, Eof, Eoln, Exp, Ln, Odd, Read, Readln, Sqr, Sqrt, Write, Writeln`. Sự khác nhau giữa tên chuẩn và từ khoá là ở chỗ người lập trình có thể định nghĩa lại tên chuẩn để dùng vào việc khác nếu muốn, còn từ khoá phải dùng đúng quy định của TP. TP không phân biệt ký tự viết thường hay viết hoa trong các từ khoá, tên hay tên chuẩn.

3. Các kiểu dữ liệu, hằng, biến

a) Các kiểu dữ liệu. Các dữ liệu (ban đầu và trung gian) dùng trong chương trình đều phải thuộc một kiểu nhất định, không thể dùng lẫn. TP có các kiểu dữ liệu chuẩn sau:

• **Kiểu số nguyên**. Kiểu số nguyên thường dùng là Integer, nó biểu diễn các số từ -32768 đến 32767, chiếm 2 byte bộ nhớ. Ngoài ra còn có các kiểu: Byte (0 đến 255, chiếm 1 byte), Shortint (-128 đến 127, chiếm 1 byte), Word (0 đến 65535, chiếm 2 byte), Longint (-2147483648 đến 2147483647, chiếm 4 byte).

• **Kiểu số thực** thường dùng: Real và Double . Kiểu Real biểu diễn các số thực có trị tuyệt đối từ $2.9*10^{-39}$ đến $1.7*10^{38}$, kiểu Real chiếm 6 byte bộ nhớ. Kiểu Double biểu diễn các số thực có trị tuyệt đối từ 5.0E-324 đến 1.7E+308, kiểu Double chiếm 8 byte bộ nhớ. Các số thực được viết dưới hai dạng: dạng dấu phẩy tĩnh (ví dụ 123.45678), dạng dấu phẩy động (ví dụ 1.23456E+2).

• **Kiểu Boolean**. Kiểu này chỉ có hai giá trị là True (đúng) và False (sai) và chiếm 1 byte bộ nhớ. Giá trị False được xem là nhỏ hơn True.

• **Kiểu char**. Mỗi giá trị kiểu Char (ký tự) chiếm 1 byte và biểu diễn một ký tự trong bảng mã ASCII. Mỗi ký tự (ví dụ chữ A, mã là 65) có thể dùng một trong ba cách để biểu diễn: 'A', chr(65), #65.

• **Kiểu String** (xâu ký tự) là một dãy ký tự bất kỳ đặt trong hai dấu nháy đơn, số ký tự không quá 255. Xâu rỗng '' là xâu không có ký tự nào. Một biến string được cấp

một số byte bằng độ dài của nó cộng thêm 1: byte đầu tiên dùng để ghi số ký tự đang được lưu trữ, mỗi byte còn lại chứa một ký tự.

b) Hằng. Hằng là một đại lượng có giá trị xác định và không thay đổi trong toàn bộ chương trình. Hằng cũng có các kiểu như kiểu dữ liệu đã mô tả ở trên: hằng số nguyên, hằng số thực, hằng ký tự, hằng xâu ký tự và hằng boolean. Để khai báo hằng ta dùng lệnh:

const tên_hằng = giá trị của hằng hoặc biểu thức hằng.

Ví dụ
```
const A=3; B=(9*3)/4; C='T'; D=True; Ho='Le Van';
```

c) Biến. Biến là đại lượng có giá trị thay đổi trong chương trình. Biến là tên một vùng nhớ lưu trữ dữ liệu. Tên biến đặt theo quy tắc đặt tên ở Mục 2. Mọi biến đều phải khai báo trước khi dùng. Cách khai báo biến:

Var dãy tên biến: kiểu dữ liệu;

Ví dụ:
```
var m,n:integer; a,b,c:real; tt:boolean; ten:string[20];
```

4. Câu lệnh

Một chương trình bao gồm nhiều câu lệnh, mỗi câu lệnh thực hiện một công việc nào đó. Trên một dòng có thể viết một hay nhiều câu lệnh. Các câu lệnh được ngăn cách nhau bởi dấu chấm phẩy (;)

Câu lệnh được chia thành hai loại: câu lệnh đơn giản và câu lệnh có cấu trúc. Câu lệnh đơn giản chỉ gồm một lệnh duy nhất như lệnh gán, lệnh nhập hay xuất dữ liệu, lệnh gọi thủ tục hay hàm, lệnh goto. Câu lệnh có cấu trúc có thể là lệnh ghép gồm một nhóm các lệnh đặt giữa begin và end, if, case, for, repeat, while. Chương trình hoạt động theo cấu trúc tuần tự, nghĩa là thực hiện lần lượt từng lệnh một, từ lệnh đầu đến lệnh cuối theo thứ tự các lệnh đã liệt kê trong chương trình.

5. Cấu trúc của một chương trình Pascal

Nói chung một chương trình Pascal gồm ba phần như sau:

- **Phần đầu đề** giới thiệu tên của chương trình: Program Tên_chương_trình;

- **Phần khai báo** mô tả các đối tượng, các kiểu dữ liệu dùng trong chương trình:
Uses . . . khai báo các Unit
Label . . . khai báo các nhãn
Const . . . khai báo các hằng
Type . . . khai báo các kiểu dữ liệu mới
Var . . . khai báo các biến
Function . . . khai báo các hàm
Procedure . . . khai báo các thủ tục

- **Phần thân chương trình** chứa các lệnh của máy tính thực hiện. Phần này được viết kẹp giữa hai từ khoá Begin và End (sau chữ End có dấu chấm .) và thường gồm 4 nội dung sau: các lệnh in quảng cáo về chương trình, các lệnh nhập dữ liệu ban đầu, các lệnh tính toán để giải quyết bài toán, các lệnh in kết quả tính toán.

Phần đầu đề và phần khai báo có thể không có nhưng phần thân chương trình nhất thiết phải có trong một chương trình. Lời giải thích trong chương trình được đặt giữa hai dấu { } hoặc (* *)

6. Biểu thức, các phép toán số học, lệnh gán

Biểu thức dùng để thể hiện một công thức toán học. Mỗi biểu thức gồm các phép toán và các toán hạng. Toán hạng có thể là hằng, biến, phần tử mảng hay hàm. Các phép toán có thể là phép toán số học, phép toán so sánh hay phép toán logic. Có thể dùng các dấu ngoặc tròn để diễn đạt thứ tự thực hiện các phép toán khi viết biểu thức.

Giá trị của biểu thức thu được khi thực hiện các phép toán trong biểu thức cũng thuộc một kiểu dữ liệu nào đó. Kiểu này do các phép toán trong biểu thức quyết định. Ví dụ, cộng trừ nhân chia hai số thực cho kết quả là một số thực; phép toán logic hay phép toán so sánh cho kết quả kiểu boolean . . . Biểu thức số học là biểu thức có giá trị kiểu byte, word, integer hoặc real.

Các phép toán số học : cộng, trừ, nhân, chia được ký hiệu là + - * /. Ngoài ra còn có phép chia nguyên DIV và phép lấy phần dư MOD. Giả sử m, n là hai số nguyên, khi đó `m div n` là phần nguyên của m chia cho n, còn `m mod n` là phần dư của phép chia m cho n. Ví dụ `9 div 2 = 4, 9 mod 2 = 1`.

Ví dụ `x=(a+b)*c/d; x1=(-b+sqrt(delta))/(2*a)`

Các hàm số học thường dùng

`abs(x)` giá trị tuyệt đối của x

`exp(x)` hàm e^x

`ln(x)` hàm loga cơ số tự nhiên của x

`sqr(x)` hàm bình phương của x

`sqrt(x)` hàm căn bậc hai của x

`sin(x)` và `cos(x)` : hàm sin và cos của x với x tính bằng radian

`int(x)` cho giá trị thực bằng phần nguyên của x

`frac(x)` cho giá trị thực bằng phần lẻ của x

`round(x)` cho gia trị nguyên bằng cách quy tròn x

`trunc(x)` cho giá trị nguyên bằng phần nguyên của x

`odd(n)` cho giá trị True nếu n lẻ, False nếu n chẵn.

Trong Pascal không có hàm tính $\log_a(x)$ và a^x. Khi cần tính các hàm này ta áp dụng các công thức: $\log_a(x) = \ln(x) / \ln(a)$ (a khác 1) và $a^x = \exp(x*\ln(a))$

Lệnh gán. Lệnh gán dùng để gán giá trị của một biểu thức, một hằng . . . cho một biến, một phần tử của mảng hay một hàm. Lệnh gán có dạng:

tên biến := biểu thức ;

Vế trái của lệnh gán chỉ có thể là tên biến, tên hàm hay tên phần tử của mảng. Khi dùng lệnh gán thì kiểu của biến và kiểu của biểu thức phải trùng nhau, trừ trường hợp biến thực (vế trái) có thể nhận giá trị nguyên (vế phải).

Ví dụ sau khi khai báo `var a,b: char; m,n: integer; x,y: real;` ta có thể thực hiện các phép gán sau:

```
a:='H'; b:=chr(36); m:=(15-7)div3; n:=trunc(10/3); x:=5.6; y:= 2.3;
```

7. Nhập xuất dữ liệu

Lệnh Readln(a1, a2, . . ., an), trong đó a1, a2, . . ., an là các biến kiểu integer, real, char hay string, dùng để đưa dữ liệu số hay ký tự từ bàn phím vào các biến a1, a2, . . ., an. Khi thực hiện lệnh này máy tính sẽ dừng lại, chờ người sử dụng đưa vào từ bàn phím đủ n dữ liệu phù hợp với kiểu của n biến tương ứng. Các dữ liệu cách nhau ít nhất một dấu cách, nhập xong ta ấn phím Enter để báo cho máy tính thực hiện lệnh. Thực hiện xong, lệnh Readln sẽ chuyển con trỏ xuống đầu dòng sau.

Lệnh Read(a1, a2, . . ., an) cũng tương tự như lệnh Readln, nhưng khi nhập xong dữ liệu cho các biến lệnh Read không chuyển con trỏ xuống đầu dòng sau.

Lệnh Readln(không kèm theo tên biến) có tác dụng tạm dừng chương trình để người sử dụng xem các thông báo do chương trình đưa ra trên màn hình, muốn chương trình chạy tiếp ta ấn phím Enter.

Lệnh Writeln(bt1, bt2, . . ., btn) sẽ in giá trị các bt1, bt2, . . ., btn trên một dòng màn hình bắt đầu từ vị trí hiện tại của con trỏ, sau đó đưa con trỏ về đầu dòng tiếp theo.

Lệnh Write(bt1, bt2, . . ., btn) tương tự lệnh writeln nhưng con trỏ không về đầu dòng tiếp theo mà vẫn đặt ở dòng hiện tại ngay sau giá trị của biểu thức cuối cùng. Một lệnh nhập dữ liệu Readln thường dùng kèm với lệnh Write để thông báo biến cần nhập.

Lệnh Writeln (không có tham số) dùng để đưa con trỏ về đầu dòng tiếp theo (xuống dòng). Có thể dùng nhiều lệnh Write để in giá trị của nhiều biến trên một dòng, cuối cùng dùng lệnh Writeln để xuống dòng.

Có hai dạng viết trong các lệnh Write và Writeln là viết không định dạng và viết có định dạng. Trong cách viết không định dạng: số nguyên và các ký tự được viết ra một cách bình thường, riêng số thực được viết ra ở dạng dấu phẩy động (ví dụ 1.23456E+02).

Trong cách viết có định dạng: mỗi số hoặc ký tự đều có qui định trước khoảng trống dành để in nó. Giả sử n là một số nguyên hay biến ký tự (char), khi đó lệnh `write('n = ',n:3)` sẽ dành 3 khoảng trống trên màn hình để in n. Giả sử x là biến thực, lệnh `write('x= ',x:13:5)` sẽ dành 13 khoảng trống trên màn hình để in biến x dưới dạng dấu phẩy tĩnh với 5 số lẻ. Các số 3, 13, 5 ở trên được thay đổi tuỳ theo yêu cầu in. Lệnh `write('x= ',x:1:5)` sẽ in biến x dưới dạng dấu phẩy tĩnh với 5 số lẻ, máy sẽ dành số ký tự vừa đủ để in toàn bộ số x.

Chương trình **TLuong.pas** (tính lương) minh hoạ cách nhập và in dữ liệu kiểu integer, real, char, string và boolean.

```
program Tinh_luong;
uses crt;
var hoten: string[30]; ngaycong: integer; luongngay: real;
    loai,ch: char; giadinh: boolean;

begin clrscr;
write('Vao ho ten nhan vien : '); readln(hoten);
write('Vao so ngay cong trong thang va luong mot ngay: ');
readln(ngaycong, luongngay);
```

```
write('Vao loai lao dong la A, B hay C : '); readln(loai);
write('Co gia dinh chua C/K ? '); readln(ch);
if (ch='C') or (ch='c') then giadinh := true else giadinh := false;
writeln; writeln('============================');
writeln('Ho va ten nhan vien : ',hoten);
write('So ngay cong : ',ngaycong:2);
write(', Luong mot ngay : ',luongngay:6:2,' do la');
writeln(', Luong thang : ',ngaycong*luongngay:7:2,' do la');
writeln('Loai lao dong : ',loai);
write('Co gia dinh chua : ');
if giadinh then writeln('co') else writeln('khong');
readln;
end.
```

Màn hình nhập dữ liệu khi chạy chương trình:

Vào họ và tên nhân viên : Bui The Tam ↵

Vào số ngày công trong tháng và lương một ngày: 26 9.75 ↵

Vào loại lao động là A, B hay C : C ↵

Có gia đình chưa C/K ? C ↵

Bài tập

1. Tính chu vi và diện tích tam giác với toạ độ ba đỉnh nhập vào từ bàn phím là (x1, y1), (x2, y2) và (x3, y3).

Hướng dẫn: công thức tính diện tích tam giác S = sqrt(p(p-a)(p-b)(p-c)), p = (a+b+c)/2 trong đó a, b, c là độ dài 3 cạnh tam giác. Độ dài nối 2 đỉnh đầu tiên là a = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2)).

2. Một người có a đồng, lãi suất gửi ngân hàng là s % một tháng. Hỏi sau t tháng người đó được cả vốn và lãi là bao nhiêu, công thức tính: exp (ln (a) + t * ln (1 + s / 100)). Chạy chương trình với a = 150 triệu đồng, s = 0.69 %, t = 24 tháng.

3. Hãy in lên màn hình tất cả các ký tự của bảng mã ASCII (mã từ 0 đến 255) bằng lệnh dạng: `writeln('65.',#65,' 219.',#219)`. Kết thúc chương trình tạo 50 tiếng bíp (có mã là 7).

4. Trong môi trường Turbo Pascal để tạo ký tự ■ chỉ cần ấn Alt + 2 1 9 (các chữ số gõ ở khu vực phía phải bàn phím). Lập chương trình dùng 8 lệnh writeln in lên màn hình chữ HA NOI cỡ to bằng cách dùng ký tự trên.

5. Viết chương trình in ra giá trị các hàm lượng giác (sin, cos, tang, cotang) của một góc từ 0 đến 360 độ được nhập từ bàn phím (góc có số đo theo số độ, số phút, số giây).

6. Viết chương trình tính tổng các chữ số của một số có 5 chữ số. Ví dụ nếu số là 73645 thì tổng là 25.

Chương 2

Các lệnh rẽ nhánh và lặp

1. Biểu thức logic, phép toán logic

Biểu thức logic là biểu thức có giá trị True hoặc False. Trong biểu thức logic thường gặp các phép toán sau:

Các phép toán so sánh có giá trị True hoặc False tuỳ theo quan hệ so sánh đó là đúng hay sai. Các ký hiệu: = bằng nhau, <> khác nhau, < nhỏ hơn, > lớn hơn, <= nhỏ hơn hoặc bằng, >= lớn hơn hoặc bằng. Hai vế của phép toán so sánh hoặc cùng có giá trị số, hoặc cùng có giá trị boolean, hoặc cùng có giá trị ký tự. Ví dụ: 7 >= 2 cho True, 10 < 5 cho giá trị False, 'B' < 'T' cho True.

Các phép toán logic thực hiện trên các biến hay giá trị boolean và cho kết quả boolean: NOT phép phủ định, AND phép và, OR phép hoặc, XOR phép hoặc loại trừ. Bảng sau cho ý nghĩa của các phép toán này.

X	Y	X and Y	X or Y	not X	X xor Y
F	F	F	F	T	F
F	T	F	T		T
T	F	F	T	F	T
T	T	T	T		F

Ví dụ biểu thức `(3+2>4) and not (true or (5-3=8))` cho False.

2. Lệnh rẽ hai nhánh if. Cú pháp lệnh:

if (biểu thức logic) then Lệnh 1 else Lệnh 2 ;

Nếu **biểu thức logic** có giá trị True thì máy thực hiện **Lệnh 1**, nếu nó có giá trị False thì máy thực hiện **Lệnh 2**. **Lệnh 1** và **Lệnh 2** có thể là lệnh đơn giản hoặc lệnh ghép (nhiều lệnh viết giữa hai từ khoá Begin và End). Trước từ khoá **else** không được viết dấu chấm phẩy. Phần **else Lệnh 2** có thể không có, trong trường hợp này nếu **biểu thức logic** là False thì máy bỏ qua lệnh **if**. Các lệnh **if** có thể lồng nhau.

Chương trình **Ifelse.pas**: nhập một số nguyên, xác định số là dương, bằng không hay âm.

```
var m: integer;
begin write('Vao so nguyen m = '); readln(m);
if m>0 then writeln('So vua nhap la duong')
       else if m=0 then writeln('So vua nhap bang khong')
            else writeln('So vua nhap la am');
readln; end.
```

Chương trình **Ptbh.pas**: giải phương trình bậc hai $ax^2 + bx + c = 0$ (a khác 0).

```
var a, b, c, d, t, x1, x2: real;
begin write('Vao a,b,c = '); readln(a,b,c); d := b*b - 4*a*c;
if abs(d)<0.0001 then writeln('Nghiem kep = ',-b/(2*a):1:5)
else if d<0 then writeln('Vo nghiem')
```

```
     else begin t := sqrt(d); writeln('x1 = ',(-b+t)/(2*a):1:5);
                writeln('x2 = ',(-b-t)/(2*a):1:5); end;
readln; end.
```

3. Kiểu liệt kê và kiểu đoạn con

Turbo Pascal có 4 kiểu dữ liệu chính, mỗi kiểu này lại có thể chia thành một số kiểu nhỏ.

• **Kiểu vô hướng** (scalar type):

- Kiểu cơ sở gồm có: kiểu logic (boolean), kiểu số nguyên (integer), kiểu số thực (real), kiểu ký tự (char).

- Kiểu vô hướng do người lập trình định nghĩa: kiểu liệt kê (enumerated), kiểu đoạn con.

• **Kiểu xâu ký tự** (string type)

• **Kiểu dữ liệu có cấu trúc** (structured type) gồm có: kiểu mảng (array), kiểu tập hợp (set), kiểu bản ghi (record), kiểu tệp (file).

• **Kiểu con trỏ** (pointer type).

Dưới đây sẽ nghiên cứu kiểu liệt kê và kiểu đoạn con.

Kiểu liệt kê

Kiểu này được định nghĩa bằng cách liệt kê tất cả các giá trị của kiểu thông qua các tên do người lập trình đặt ra và danh sách các giá trị tên được đặt trong hai dấu ngoặc đơn. Khi đó các giá trị trong danh sách được xem là hằng. Thứ tự các phần tử liệt kê trong danh sách đánh số từ 0 cho đến hết. Các giá trị kiểu liệt kê có thể so sánh với nhau theo qui định: giá trị đứng trước nhỏ hơn giá trị đứng sau. Phép toán duy nhất dùng cho kiểu liệt kê là phép so sánh. Có thể thực hiện phép gán trên các kiểu liệt kê. Giá trị thuộc kiểu liệt kê thường được dùng để làm chỉ số cho vòng lặp for, các khả năng lựa chọn trong lệnh Case, chỉ số cho các mảng. Không thể dùng các lệnh Readln, Writeln với dữ liệu kiểu liệt kê.

Đối với kiểu liệt kê, hàm Ord chuyển một giá trị kiểu liệt kê sang số nguyên (là số thứ tự của giá trị này trong định nghĩa kiểu liệt kê), hàm Pred cho giá trị đứng trước của đối số, hàm Succ cho giá trị đứng sau của đối số.

Chương trình **LietKe.pas** minh hoạ cách dùng kiểu liệt kê.

```
type ngay=(ThuHai,ThuBa,ThuTu,ThuNam,ThuSau,ThuBay,ChuNhat);
var x: ngay;
begin
for x := ThuHai to ChuNhat do writeln(ord(x));  {0,1,2,3,4,5,6}
writeln(ThuBa < ThuBay,'  ',ThuTu >= ChuNhat);  {TRUE FALSE}
x := ThuNam; writeln(pred(x)=ThuTu,' ',succ(x)=ThuSau); {TRUE TRUE}
readln;
end.
```

Kiểu đoạn con. Kiểu đoạn con được định nghĩa trên cơ sở các kiểu nguyên, logic, ký tự và kiểu liệt kê bằng cách giới hạn chỉ xét các giá trị của nó nằm trong khoảng một cận dưới và một cận trên. Kiểu đoạn con giúp cho chương trình dễ đọc, tiết

kiệm bộ nhớ và có thể kiểm tra xem giá trị của biến kiểu đoạn con có vượt ra ngoài miền giới hạn của nó hay không. Cách khai báo như sau:

type tên miền con = (hằng dưới . . hằng trên) ;

Ví dụ:
```
type chuso = '0'..'9'; chuhoa = 'A'..'Z'; thang = 1..12;
var s : chuso;  ch : chuhoa; t : thang;
```

4. Lệnh rẽ nhiều nhánh case of

Lệnh case of tiện dùng khi chương trình cần chia thành nhiều nhánh công việc. Cú pháp của lệnh:

Case biểu_thức of
tập hằng 1 : lệnh 1 ;
tập hằng 2 : lệnh 2 ;
. . . . . .
tập hằng n : lệnh n
Else lệnh n+1;
End ;

Tập hằng i (i=1, 2,. . ., n) là các hằng hay đoạn hằng cùng một kiểu (nguyên, ký tự, logic, liệt kê). Ví dụ **tập hằng i** có thể là: 5, 10..15, '0'..'9', 't', chr(150). Giá trị của **biểu thức** và giá trị trong các **tập hằng i** phải có cùng một kiểu. Phần **else lệnh n+1** có thể không có, nhưng khi đó giữa **lệnh n** và **End** phải có dấu chấm phẩy.

Sự hoạt động của lệnh: trước tiên máy xác định giá trị của **biểu thức**. Nếu **tập hằng i** là tập đầu tiên chứa giá trị của **biểu thức** thì máy thực hiện **lệnh i** và thoát khỏi lệnh case. Trong trường hợp không có **tập hằng i** nào chứa giá trị của **biểu thức** thì máy sẽ thực hiện **lệnh n+1** nếu có dùng **else** hoặc thoát khỏi lệnh **case** nếu không dùng **else**. **Lệnh i** ở trên có thể là một lệnh đơn giản hoặc một lệnh ghép.

Chương trình **Tuoi.pas** minh hoạ cách dùng lệnh case để rẽ nhiều nhánh:

```
var tuoi : byte;
begin write('Xin cho biet tuoi cua ban : '); readln(tuoi);
case tuoi of
       1..6   : writeln('Tuoi nho');
       7..15  : writeln('Tuoi thieu nhi');
       16..30 : writeln('Tuoi thanh nien');
       31..60 : writeln('Tuoi trung nien')
       else writeln('Tuoi gia');
end;
readln;  end.
```

5. Lệnh lặp for

Khi lập trình chúng ta thường gặp nhiều trường hợp phải lặp đi lặp lại một số công việc nào đó. Số lần lặp có thể xác định hay không xác định. Nếu số lần lặp xác định ta nên dùng lệnh For, nếu số lần lặp không xác định nên dùng lệnh repeat hay while.

Lệnh for có hai dạng là dạng tiến và dạng lùi:

for b := bt1 to bt2 do Lệnh ; (1)

for b := bt1 downto bt2 do Lệnh ; (2)

trong đó b là biến đếm, bt1 và bt2 là các biểu thức, cả ba cần có cùng một kiểu giá trị (byte, word, integer, char, boolean hay kiểu liệt kê).

Sự hoạt động của dạng (1): đầu tiên **b** nhận giá trị **bt1**, máy kiểm tra điều kiện ord(b) <= ord(bt2). Nếu điều kiện này là sai thì máy ra khỏi vòng lặp **for** và thực hiện lệnh tiếp theo trong chương trình. Nếu điều kiện trên là đúng thì máy thực hiện **Lệnh**, đồng thời biến **b** được tăng tới giá trị tiếp theo succ(b), máy lại kiểm tra điều kiện ord(b) <= ord(bt2), quá trình tiếp tục.

Sự hoạt động của dạng (2) tương tự dạng (1), chỉ khác ở chỗ: điều kiện kiểm tra bây giờ là ord(b) >= ord(bt2) và khi điều kiện đúng thì máy thực hiện **Lệnh** và biến **b** giảm đến giá trị đứng ngay trước pred(b).

Thông thường **Lệnh** sau **do** là một câu lệnh ghép. Khi dùng lệnh for cần lưu ý: không được thay đổi trị của biến đếm **b** bằng một lệnh trong vòng lặp và khi ra khỏi vòng lặp giá trị của biến đếm là không xác định.

Chương trình **Ascii.pas**: in 256 ký tự của bảng mã ASCII.

```
uses crt;
var i : byte;
begin clrscr;
for i := 0 to 255 do
      begin writeln('Ky tu thu ',i,' la : ',chr(i));
            if (i mod 21)=0 then readln;
      end;
readln; end.
```

Chương trình **Tong.pas**: tính tổng n số thực nhập từ bàn phím.

```
uses crt;
var i,n : integer; a,s : real;
begin clrscr;
write('So luong so n = '); readln(n); s := 0;
for i:=1 to n do
   begin write('a[',i,'] = '); readln(a);
         s := s + a;
   end;
writeln('Trung binh cong = ',s/n:13:5);
readln; end.
```

Chương trình **ChuCai.pas**: in 4 dòng chữ cái từ a tới z, in thường và in hoa theo cả hai chiều tiến và lùi.

```
uses crt;
var ch : char;
begin clrscr;
for ch := 'a' to 'z' do write(ch,' '); writeln;
for ch := 'z' downto 'a' do write(ch,' '); writeln;
for ch := 'A' to 'Z' do write(ch,' '); writeln;
for ch := 'Z' downto 'A' do write(ch,' '); writeln;
readln;
end.
```

6. Lệnh while . . . do

Cú pháp lệnh : **while biểu_thức_logic do Lệnh ;**

Hoạt động của lệnh: máy sẽ lặp đi lặp lại **Lệnh** chừng nào **biểu thức logic** còn nhận giá trị true. Nếu **biểu thức logic** ngay từ đầu đã nhận giá trị false thì máy không thực hiện gì và chuyển tới lệnh tiếp theo. Thông thường **Lệnh** sau **do** là một lệnh ghép.

Chương trình **While.pas**: tìm ước chung lớn nhất và bội số chung nhỏ nhất của hai số nguyên dương m và n nhập vào từ bàn phím.

```
uses crt;
var m, n, bsc: integer;
begin clrscr;
write('Vao hai so nguyen m, n = '); readln(m,n);
bsc := m * n;
while m<>n do if m>n then m:=m-n else n:=n-m;
writeln('Uoc so chung lon nhat = ',m);
writeln('Boi so chung nho nhat = ',bsc div m);
readln; end.
```

7. Lệnh repeat . . . until

Cú pháp lệnh: **repeat Lệnh until biểu_thức_logic ;**

Hoạt động của lệnh: đầu tiên máy thực hiện **Lệnh**, sau đó kiểm tra **biểu thức logic**; nếu **biểu thức logic** có giá trị true thì máy ra khỏi vòng lặp, còn nếu nó có giá trị false thì máy lại quay lên thực hiện **Lệnh**. Thông thường **Lệnh** gồm một dãy các lệnh liên tiếp, nhưng không cần đặt giữa begin và end. Thân vòng lặp (**Lệnh**) được thực hiện ít nhất một lần.

Chương trình **Repeat.pas**: nhập các ký tự cho đến khi nhập 'X' thì kết thúc.

```
uses crt;
var c: char;
begin clrscr;
writeln('Vao cac ky tu, an X de thoat');
repeat c := readkey;
       write(c,' ');
until c='X';
end.
```

8. Các lệnh goto, break, exit, halt

Lệnh goto có cú pháp: **goto nh;**

trong đó **nh** là một nhãn. Nhãn có thể là một tên như tên biến hoặc là một số nguyên từ 0 đến 9999. Tên nhãn phải được khai báo ở đầu chương trình bằng lệnh **label**. Khi gặp lệnh **goto nh** máy sẽ nhảy không điều kiện tới thực hiện câu lệnh có nhãn là **nh**.

Lệnh goto chỉ cho phép nhảy từ vị trí này tới vị trí khác trong thân một hàm hay thủ tục, cho phép nhảy từ trong một vòng lặp ra ngoài; không cho phép nhảy từ ngoài vào một hàm hay thủ tục, không cho phép nhảy vào trong một vòng lặp, không cho phép nhảy từ ngoài vào một khối lệnh. Bằng cách nhảy ngược trở lại, lệnh goto cho phép tạo được các chu trình như lệnh for.

Chương trình **LapGoto.pas** : tính tổng 1 + 2 + 3 + . . . + n.

```
label Nhan;
var i,s,n: integer;
begin   write('Vao n = ');  readln(n);
s:=0; i:=1;
Nhan : s:=s+i;
       i:=i+1;
if i<=n then goto Nhan;
write('Tong = ',s);
readln; end.
```

Lệnh Break. Trong thân các lệnh chu trình For, While, Repeat khi gặp lệnh Break thì máy sẽ thoát khỏi chu trình. Nếu có nhiều chu trình lồng nhau máy chỉ thoát ra khỏi chu trình trong nhất chứa lệnh Break.

Chương trình **Break.pas**: tính tổng các số nguyên nhập vào từ bàn phím, nếu số nhập vào là số 0 thì kết thúc chương trình.

```
uses crt;
var i,s: integer;
begin clrscr; s := 0;
while true do
  begin  write('Vao mot so, nhap 0 ket thuc : ');
         readln(i); if i=0 then break;
         s := s + i;
  end;
writeln('Tong cac so da nhap : ',s);
readln; end.
```

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

Lệnh Halt. Lệnh Halt làm dừng hẳn chương trình, thường dùng khi gặp phải một trường hợp nào đó mà thuật toán không thể tiếp tục được.

Bài tập

1. Lập trình vào 3 số thực a, b, c. Kiểm tra xem 3 số đó có thể là độ dài 3 cạnh của một tam giác hay không. Nếu chúng là 3 cạnh của một tam giác thì kiểm tra xem đó là tam giác gì (vuông, cân, đều) và tính diện tích tam giác.

2. Giải bất phương trình bậc hai $AX^2 + BX + C > 0$ với các hệ số A, B, C bất kỳ nhập vào từ bàn phím.

3. Giải phương trình $AX^4 + BX^2 + C = 0$ với các hệ số nhập từ bàn phím

4. Lập trình giải hệ phương trình: $AX + BY = C$

$DX + EY = F$

Hướng dẫn: tính dt = ae-db, dx = ce-fb, dy = af-dc. Nếu dt<>0 thì có duy nhất nghiệm x = dx/dt, y = dy/dt. Trong trường hợp dt = 0, nếu dx = 0 và dy = 0 thì hệ phương trình có vô số nghiệm, trái lại thì hệ phương trình vô nghiệm.

5. Viết chương trình nhập vào số thứ tự của Tháng và Năm (chẳng hạn tháng 9/2004: nhập 9 và 2004), in ra màn hình số ngày của tháng đó. Lưu ý: các tháng 1, 3,

5, 7, 8, 10, 12 có 31 ngày; các tháng 4, 6, 9, 11 có 30 ngày; tháng 2 có 28 ngày, vào năm nhuận (năm chia hết cho 4) tháng 2 có 29 ngày.

6. Lập chương trình đọc vào từ bàn phím một số nguyên N (1<= N <= 10) rồi đưa ra tiếng Anh của số đó. Ví dụ: gõ vào 2, đưa ra TWO.

7. Nhập vào từ bàn phím một ký tự, thông báo ra màn hình ký tự đó thuộc vào tập hợp các ký tự nào trong 8 tập ký tự: các chữ số, chữ thường, chữ hoa, các dấu phép tính, các dấu so sánh, các loại dấu ngoặc, các dấu ngắt câu, các ký tự khác.

8. Tìm tất cả các số nguyên tố nằm giữa hai số tự nhiên N1 và N2 (hai số này nhập vào từ bàn phím).

9. Tìm tất cả các số tự nhiên không lớn hơn 10000 sao cho nó bằng tổng các ước của nó, ước không kể chính nó. Ví dụ: 28 = 1 + 2 + 4 + 7 + 14.

10. Hãy liệt kê Kim phút gặp Kim giờ vào những thời điểm nào kể từ 0 giờ cho tới 24 giờ cùng ngày, cho tổng số lần gặp nhau (kể cả lúc 0 giờ và 24 giờ).

11. Viết chương trình nhập vào một số nguyên dương (kiểu long) từ bàn phím và in ra màn hình số lượng chữ số của số vừa nhập. Ví dụ: 789123456 cho 9.

12. Tính xấp xỉ hàm e^x theo công thức $e^x = 1 + (x / 1!) + (x^2 / 2!) + \ldots + (x^n / n!)$, trong đó x được cho từ bàn phím. Tổng trên chỉ giữ lại các số hạng có trị tuyệt đối lớn hơn một số eps cho trước, eps cũng được nhập vào từ bàn phím.

13. Hàm sin(x) được tính xấp xỉ theo công thức

$$Sin(x) = x - x^3 / 3! + x^5 / 5! - x^7 / 7! + x^9 / 9! - \ldots .$$

với x tính theo radian. Hãy viết chương trình tính giá trị với góc x nhập vào từ bàn phím theo số độ, số phút và số giây. Tổng trên chỉ giữ lại các số hạng có trị tuyệt đối lớn hơn 0.00001. So sánh kết quả với hàm chuẩn sin(x) của Turbo Pascal.

14. Tính căn bậc hai của một số t dương theo công thức lặp $x_n=(t / x_{n-1} + x_{n-1}) / 2$
Giá trị xấp xỉ ban đầu là số dương x_0 cho trước. Tiêu chuẩn dừng của quá trình lặp là $abs(x_n - x_{n-1}) < 0.00001$.

15. Lập trình làm nhiệm vụ : khởi tạo số dư tài khoản là 0, hiện menu sau cho phép người dùng lựa chọn một trong 4 khả năng:

X. Xem tài khoản G. Gửi tiền vào R. Rút tiền ra K. Kết thúc

Chương trình sẽ chạy theo vòng lặp và kết thúc khi người dùng ấn 'K'. Nếu chọn 'G', máy sẽ hiện dòng chữ hỏi "Gửi vào bao nhiêu tiền ? ", người dùng nhập vào số lượng tiền gửi vào ngân hàng, máy tính toán và hiện "Số dư tài khoản mới : . . . . ". Nếu chọn 'R', máy sẽ hiện dòng chữ hỏi "Rút ra bao nhiêu tiền ? ", người dùng nhập vào số lượng tiền cần rút, máy tính toán và hiện "Số dư tài khoản mới : . . . . ". Nếu chọn 'X' máy sẽ hiện "Số dư hiện tại là . . .". Yêu cầu kiểm tra: số dư tài khoản, số tiền gửi vào và rút ra không thể âm.

Chương 3

Kiểu mảng, kiểu xâu ký tự, kiểu tập hợp

1. Kiểu mảng

Mảng là một kiểu dữ liệu có cấu trúc bao gồm một số cố định các phần tử có cùng kiểu, có cùng một tên chung. Mỗi phần tử của mảng chứa được một giá trị và được truy xuất thông qua các chỉ số. Mảng cũng có các kiểu như biến: byte, integer, real, char, string . . . Công dụng của mảng là dùng để lưu trữ một dãy số liệu có cùng một tính chất nào đó, ví dụ mảng họ tên các sinh viên.

a) Mảng một chiều

Mảng một chiều là mảng dùng một chỉ số để xác định các phần tử của mảng. Trước khi dùng mảng phải khai báo kiểu của nó. Có 3 cách khai báo mảng một chiều:

Cách 1: khai báo qua định nghĩa kiểu của mảng

```
const nmax = 3000;
type mang1 = array[1..nmax] of real;
     mang2 = array['a'..'z'] of integer;
     mang3 = array[0..99] of string[25];
var a : mang1; b : mang 2; s : mang3;
```

Lệnh type định nghĩa kiểu của mảng, bao gồm: tên kiểu, giới hạn dưới và giới hạn trên của chỉ số chạy trong mảng, kiểu giá trị của các phần tử của mảng. Các lệnh trên khai báo: biến **a** là một mảng số thực có tối đa 3000 phần tử, biến **b** là mảng số nguyên có tối đa 26 phần tử (đánh số các phần tử từ a đến z), biến s là một mảng xâu ký tự có tối đa 100 phần tử và mỗi xâu ký tự dài không quá 25 ký tự. Để truy nhập vào phần tử thứ i của mảng a ta dùng cách viết **a[i]**.

Cách 2: khai báo trực tiếp

```
var a : array[1..1000] of double ;
```

Cách 3: khai báo và khởi đầu cho mảng

```
const a : array[1..6] of integer = (5, -1, 7, 8, -9, 6);
      tamgiac : array[1..4] of pointtype =
          ((x:320;y:120),(x:480;y:360),(x:160;y:240),(x:320;y:120));
```

Lệnh đầu khai báo mảng **a** gồm 6 phần tử và gán giá trị ban đầu, trong chương trình các a[i] có thể thay đổi giá trị mặc dù có chữ const. Lệnh thứ hai khai báo mảng **tamgiac** có 4 phần tử và gán giá trị ban đầu, một phần tử là một điểm trên mặt phẳng (có kiểu bản ghi).

Chú ý chỉ số chạy trong mảng có thể là các số nguyên âm và mảng dùng trong các chương trình con (thủ tục hay hàm) đều phải khai theo cách thứ nhất.

Chương trình **NoiBot.pas**: nhập một mảng n số thực từ bàn phím, dùng thuật toán nổi bọt để sắp xếp các phần tử của mảng theo chiều tăng dần. Sau mỗi vòng lặp theo i phần tử lớn nhất chìm xuống cuối cùng, chương trình in ra các kết quả sắp xếp trung gian.

```
uses crt;
```

```
const nmax = 100;
type mang = array[1..nmax] of real;
var a : mang; i,j,n : integer; t: real;
begin clrscr;
write('Vao so phan tu cua mang n = '); readln(n);
for i:=1 to n do begin write('a[',i,'] = '); readln(a[i]) end;
for i:=1 to n-1 do
  begin for j:=1 to n-i do if a[j]>a[j+1] then
            begin t:=a[j]; a[j]:=a[j+1]; a[j+1]:=t end;
        writeln('Buoc lap i = ',i);
        for j:=1 to n do write(a[j]:8:2,'  '); writeln;
  end;
writeln('Ket qua sap xep:');
for j:=1 to n do write(a[j]:8:2,'  '); writeln;
readln;
end.
```

Chương trình **NguyenTo.pas**: tìm các số nguyên tố không vượt quá n cho trước.

```
uses crt;
const n = 9999;
var a: array[1..n] of boolean; i,j,d: integer;
begin clrscr; a[1]:= false; d:=0;
for i:=2 to n do a[i] := true;
for i:=2 to n div 2 do for j:=2 to n div i do a[i*j]:= false;
for i:=1 to n do if a[i] then
  begin d := d+1; write(i:5); if d mod (24*16) =0 then readln; end;
readln;
end.
```

b) Mảng hai chiều

Trong thực tế chúng ta thường phải làm việc với các bảng số liệu gồm các hàng và cột (ví dụ bảng điểm của học sinh). Để biểu thị bảng số ta dùng mảng hai chiều với hai chỉ số, chỉ số đầu để chỉ hàng, chỉ số thứ hai để chỉ cột. Cách khai báo mảng hai chiều:

```
const mmax = 30; nmax = 50;
type mang = array[1..mmax, 1..nmax] of integer;
var a : mang ;
```

Các lệnh trên khai báo biến a là một mảng hai chiều các số nguyên có chỉ số hàng chạy từ 1 đến 30 và chỉ số cột chạy từ 1 đến 50. Để truy nhập vào phần tử hàng i cột j của mảng **a** ta dùng cách viết **a[i,j]**. Ta cũng có thể khai báo mảng hai chiều một cách trực tiếp (không dùng type):

```
var a : array[1..30, 1..50] of integer ;
```

Đối với những chương trình cần tiết kiệm thời gian tính toán, các mảng hai chiều nên quy về một chiều. Với mảng hai chiều có 30 hàng và 50 cột có thể quy về mảng một chiều theo khai báo sau:

```
const m = 30; n = 50;
type mang = array[1..m*n] of integer;
var a : mang ;
```

Biến a là mảng một chiều có m*n phần tử. Để truy nhập tới phần tử hàng i cột j của mảng hai chiều gốc ta dùng cách viết **a[k]** với k = (i-1)*n+j.

Độ lớn tối đa của một mảng một chiều hay hai chiều không vượt quá 64 KB.

Tương tự mảng hai chiều ta có thể khai báo các mảng có nhiều hơn hai chỉ số:

```
var a : array[1..10,1..10,1..10,1..10] of integer;
```

Chương trình **Mang2c.pas**: cộng mảng hai chiều theo hàng và theo cột.

```
uses crt;
const mmax = 20; nmax = 30;
type mang = array[1..mmax, 1..nmax] of real;
var a: mang; i,j,m,n: integer; s: real;
begin clrscr; write('Vao so hang va so cot m, n = '); readln(m,n);
for i:=1 to m do for j:=1 to n do
  begin write('a[',i,',',j,'] = '); readln(a[i,j]); end;
for i:=1 to m do
  begin s:=0; for j:=1 to n do s:= s+a[i,j];
        writeln('Tong hang ',i,' la : ',s:13:5);
  end;
for j:=1 to n do
  begin s:=0; for i:=1 to m do s:= s+a[i,j];
        writeln('Tong cot ',j,' la : ',s:13:5);
  end;
readln; end.
```

2. Kiểu xâu ký tự

a) Khai báo xâu ký tự

Dữ liệu kiểu xâu ký tự (string type) là một dãy các ký tự bất kỳ. Dữ liệu kiểu xâu ký tự có nhiều điểm tương tự như kiểu mảng nhưng cũng có điểm khác nhau là số ký tự trong một biến kiểu xâu ký tự có thể thay đổi, còn số phần tử của kiểu mảng luôn luôn cố định. Một biến kiểu char là một xâu ký tự có độ dài một. Một biến kiểu xâu ký tự được khai báo bằng từ khoá string theo sau là số ký tự cực đại có thể có của xâu đặt trong hai dấu ngoặc vuông [ ]

var tên_biến : string[độ dài cực đại] ;

Xâu ký tự có chiều dài tối đa là 255. Khi khai báo ta cũng có thể không chỉ ra độ dài cực đại. Ví dụ: `Var HoTen: string[25]; St: string;` Câu lệnh này khai báo biến HoTen có thể chứa tối đa 25 ký tự, biến St chứa tối đa 255 ký tự. Để chỉ ký tự thứ i trong xâu ký tự St, ta dùng ký hiệu St[i] (giống như ký hiệu phần tử thứ i của mảng).

Cách bố trí bộ nhớ của xâu ký tự

Giả sử biến HoTen ở trên được gán giá trị `HoTen := 'Nguyen Van';` Khi đó phần tử thứ 0 của xâu ký tự HoTen[0] chứa ký tự mà mã thập phân ASCII của nó là số ký tự hiện có trong xâu ký tự (tức là **chr(10)**). Tiếp theo từ HoTen[1] đến HoTen[10] chứa các ký tự N g u y e n V a n. Các phần tử từ HoTen[11] đến HoTen[25] là không xác định. Như vậy độ dài thực của xâu ký tự bằng `ord(HoTen[0])`.

Các phép toán trên xâu ký tự

- Phép gán: **biến := biểu thức**. Ví dụ `HoTen := 'Nguyen Van Minh';`
- Phép nối xâu ký tự: ký hiệu bởi dấu cộng (+). Ví dụ `'Turbo '+'Pascal'`.

- Các phép toán so sánh: khi so sánh hai xâu ký tự, các ký tự của hai xâu được so sánh từng cặp một từ trái sang phải theo giá trị của bảng mã ASCII. Nếu hai xâu có độ dài khác nhau và số ký tự giống nhau đến độ dài xâu ngắn nhất thì xâu có độ dài ngắn hơn được xem là bé hơn. Ví dụ: 'ABC' = 'ABC' cho True, 'ABC' > 'AB' cho True, 'abcd' < 'abed' cho True, 'abc' > 'ad' cho False.

b) Các thủ tục và hàm chuẩn xử lý xâu ký tự

Turbo Pascal có sẵn nhiều thủ tục và hàm chuẩn để xử lý xâu ký tự:

Thủ tục Delete(St, Vt, Sokt). Các tham số: St là biến kiểu xâu ký tự, Vt và Sokt là các biểu thức nguyên. Thủ tục này sẽ xoá từ vị trí Vt của xâu ký tự St một số ký tự là Sokt. Ví dụ nếu St = 'abcdefgh' thì Delete(St,3,4) sẽ xoá 4 ký tự của St kể từ vị trí thứ ba, kết quả ta nhận được St = 'abgh'.

Thủ tục Insert(St2, St1, Vt). Các tham số: St2 là biểu thức kiểu xâu ký tự, St1 là biến kiểu xâu ký tự, Vt là biểu thức kiểu nguyên. Thủ tục này sẽ chèn xâu ký tự St2 vào xâu ký tự St1 kể từ vị trí Vt. Ví dụ St1 = 'Nguyen Dung', St2 = 'Van', lệnh Insert(St2, St1,8) cho ta St1 = 'Nguyen Van Dung'. Trường hợp Vt vượt quá độ dài của St1 thì St2 sẽ được ghép nối tiếp vào sau St1. Ví dụ St1 = 'abcd' thì Insert('xyz',St1,6) cho kết quả 'abcdxyz'.

Thủ tục Str(GiaTri, St), trong đó GiaTri là biểu thức nguyên hay thực có ghi dạng in ra, St là biến kiểu xâu ký tự. Thủ tục này sẽ đổi giá trị số GiaTri thành xâu ký tự rồi gán cho St. Ví dụ i = 45, Str(i:3, St) cho ta St = ' 45' (có 3 ký tự, ký tự đầu là dấu cách). Giả sử x = 123.5678999 thì Str(x:10:5, St) cho ta St = ' 123.56789'.

Thủ tục Val(St, So, Loi), trong đó St là biểu thức kiểu xâu ký tự, So là biến nguyên hay thực, Loi là biến nguyên. Thủ tục này đổi xâu ký tự St biểu diễn một số (nguyên hay thực) thành ra số và gán cho So, nếu phép biến đổi đúng thì Loi = 0, nếu phép biến đổi sai (do St không biểu diễn đúng số nguyên hay thực) thì Loi nhận giá trị bằng vị trí của ký tự sai trong xâu St. Ví dụ St = '234', k và l là hai biến nguyên thì Val(St, k, l) cho k = 234, l = 0. Giả sử St = '234a6' thì Val(St, k, l) cho k không xác định và l = 4.

Hàm Length(St) cho độ dài của xâu ký tự St. Ví dụ: nếu St = 'Ha noi' thì Length(St) = 6.

Hàm Copy(St, Vt, Sokt), trong đó St là xâu ký tự, Vt và Sokt là các biểu thức nguyên. Hàm này sao chép Sokt ký tự kể từ vị trí Vt của xâu St sang một xâu ký tự mới. Nếu Vt + Sokt > Length(St) thì Copy chỉ sao chép các ký tự nằm trong xâu st. Còn nếu Vt > Length(St) thì Copy sẽ cho một xâu ký tự rỗng. Ví dụ St = 'abcdefgh' thì St1 = Copy(St, 3, 4) sẽ cho xâu ký tự mới St1 = 'cdef'.

Hàm Concat(s1, s2,. . . , sn) sẽ ghép nối tất cả các xâu ký tự s1, s2, . . ., sn thành một xâu ký tự theo thứ tự đã viết. Nếu tổng số chiều dài của các xâu ký tự lớn hơn 255 thì máy báo lỗi. Ta cũng có thể dùng phép cộng để ghép nối các xâu ký tự.

Hàm Pos(S1, S2) cho ta vị trí đầu tiên của xâu ký tự S1 gặp trong xâu ký tự S2. Nếu không tìm thấy Pos cho giá trị bằng 0.

Hàm Upcase(ch: char) đổi chữ cái thường thành chữ cái hoa.

Chương trình **ChuChay.pas**: in lên màn hình một dòng chữ chạy từ phải sang trái cho đến khi ấn một phím bất kỳ.

```
uses crt;
var i,d: integer; s, s1, s2: string;
begin clrscr;
s := 'Trung tam Khoa hoc Tu nhien va Cong nghe Quoc gia * ';
d := length(s);
repeat gotoxy(15,11); write(s); delay(250);
s1:= copy(s,2,d-1); s2:= copy(s,1,1); s:= s1+s2;
until keypressed;
end.
```

Chương trình **XepXau.pas**: nhập danh sách học sinh một lớp, đổi các tên ra chữ hoa, sắp xếp theo thứ tự tăng dần của họ và tên.

```
uses crt;
const nmax = 100;
type mang = array[1..nmax] of string[25];
var a : mang; i,j,n : integer;
    t: string[25];
begin clrscr;
write('Vao so phan tu cua mang n = '); readln(n);
for i:=1 to n do
   begin write('a[',i,'] = '); readln(a[i]); t := a[i];
         for j:=1 to length(t) do t[j] := upcase(t[j]);
         a[i] := t
   end;
for i:=1 to n-1 do  for j:=1 to n-i do
  if a[j]>a[j+1] then
     begin t:=a[j]; a[j]:=a[j+1]; a[j+1]:=t end;
writeln('Ket qua sap xep:');
for j:=1 to n do write(a[j],' -> '); writeln;
readln;
end.
```

3. Dữ liệu kiểu tập hợp

a) Khai báo kiểu tập hợp

Dữ liệu kiểu tập hợp (Set) là một tập hợp của những dữ liệu cùng thuộc một kiểu rời rạc, gọi là kiểu cơ sở của kiểu tập hợp. Kiểu cơ sở có thể là kiểu nguyên, kiểu ký tự, kiểu logic, kiểu liệt kê hay kiểu đoạn con. Trong Turbo Pascal, số phần tử cực đại của một tập hợp là 256 phần tử. Một kiểu tập hợp được khai báo bởi:

Tên kiểu tập hợp = Set of Kiểu cơ sở;

Ví dụ:

```
type ChuSo = Set of 0..9;
     ChuCaiHoa = Set of 'A'..'Z';
var  so : ChuSo; chu : ChuCaiHoa;
     Mau : set of (Xanh, Do, Tim, vang);
```

Một tập hợp được xác định bằng cách liệt kê các phần tử của nó trong hai dấu móc vuông. Với ví dụ ở trên ta có thể thực hiện các phép gán sau: `so := [2,3,6..9]; chu := ['A','C'..'F'];` Vị trí các phần tử trong tập hợp không

có ý nghĩa. Tập hợp rỗng được ký hiệu là [] và nó được xem là cùng kiểu với mọi tập hợp.

b) Các phép toán trên tập hợp

Các phép toán quan hệ: A = B cho giá trị True nếu hai tập hợp bằng nhau, A<>B cho True nếu hai tập hợp khác nhau, A<=B cho True nếu A là tập con của B, A>=B cho True nếu B là tập con của A. Chú ý: không có phép toán > hay <. Để kiểm tra xem tập A có thực sự nằm trong tập B hay không ta sử dụng câu lệnh:

```
if (A<>B) and (A<=B) then writeln('A la tap con thuc su cua B');
```

Phép toán in. Phép toán **in** dùng để xét xem một phần tử nào đó có nằm trong tập hợp không. Nếu phần tử đó có nằm trong tập hợp thì phép toán cho giá trị True, ngược lại cho giá trị False. Ví dụ: `'c' in ['a','c','d']` cho giá trị True, `'n' in ['Y','y']` cho False.

Các phép toán hợp, giao, hiệu. A+B là hợp của A và B, A*B là giao của A và B, A-B là hiệu của A và B. Ví dụ : nếu A = [1, 3, 5, 7, 9], B = [5, 6, 7, 8] thì A+B = [1, 3, 5, 6, 7, 8, 9], A*B = [5,7], A-B = [1, 3, 9].

Chương trình **TapHop.pas**: nhập vào một chữ cái, xét xem chữ cái đó là nguyên âm hay phụ âm.

```
uses crt;
type tapkt = set of char;
var ChuCai,NguyenAm: tapkt; ch: char;
begin clrscr;
ChuCai := ['a'..'z','A'..'Z'];
NguyenAm := ['A','E','I','O','U'];
repeat write('Nhap mot chu cai : '); readln(ch);
until ch in ChuCai;
if upcase(ch) in NguyenAm then
     writeln(ch,' la mot nguyen am')
else writeln(ch,' la mot phu am');
readln;
end.
```

Bài tập

1. Nhập một dãy số gồm N phần tử từ bàn phím. Đưa ra các thông tin:
- Tổng các phần tử của dãy, trung bình cộng của các phần tử
- Số lượng các số hạng dương và tổng của chúng
- Số hạng âm đầu tiên và chỉ số của nó
- Số hạng lớn nhất của dãy và chỉ số của nó
- Số lượng số hạng dương liên tiếp nhiều nhất
- Số lượng số hạng dương liên tiếp có tổng lớn nhất
- Số lượng các số hạng liên tiếp đan dấu nhiều nhất
- Số lượng các phần tử lớn hơn hay bằng một số X cho trước

2. Nhập một dãy số nguyên từ bàn phím. In ra màn hình tần suất (số lần xuất hiện) của các số trong dãy đó.

3. Nhập từ bàn phím tuổi và thu nhập của N người (dùng hai mảng một chiều). Tính thu nhập trung bình của mỗi lứa tuổi.

4. Nhập một số nguyên dương n <= 200 và hai dãy n số nguyên a[1..n] và b[1..n]. Thông báo ra màn hình xem hai dãy đó có cùng các số hạng như nhau và chỉ khác nhau về thứ tự sắp xếp hay không ?

5. Dùng các mảng một chiều nhập dữ liệu từ bàn phím của các khách hàng gửi tiền vào ngân hàng: Họ tên, Số tài khoản, Số dư tiền gửi (có thể âm). Sắp xếp danh sách khách hàng theo số tài khoản tăng dần và in toàn bộ dữ liệu ra dưới dạng bảng, mỗi hàng là dữ liệu của một khách hàng, tính tổng số tiền dư của các khách hàng.

6. Ba đại lý A, B, C bán 10 loại báo. Viết một chương trình nhập vào tên báo, lượng bán trong một tuần của mỗi đại lý đối với loại báo này, tính tổng lượng bán của loại báo. Đưa toàn bộ thông tin ra một bảng với 5 cột: tên báo, lượng bán của từng đại lý, tổng số. Mỗi dòng ứng với một loại báo. Các mảng dùng: T(10) mảng tên các báo, A(10) lượng bán của đại lý A, B(10) lượng bán của đại lý B, C(10) lượng bán của đại lý C, S(10) tổng lượng bán của 10 loại báo. Công thức tính: S(i) = A(i) + B(i) + C(i).

7. Biết ngày 1/1/2004 là ngày thứ năm, in toàn bộ cuốn lịch năm 2004, 2005, 2006 ra màn hình (có dừng trang màn hình):

1/1/2004 Thứ năm
2/1/2004 Thứ sáu .....

8. Dùng mảng một chiều để lưu trữ các số nguyên dương lớn (có thể có hàng ngàn chữ số). Lập trình thực hiện các phép tính cộng, trừ và nhân hai số nguyên dương cỡ lớn.

9. Nhập một mảng hai chiều A (m hàng, n cột) từ bàn phím.

- Tìm giá trị lớn nhất và nhỏ nhất trên mỗi cột.
- Tìm phần tử lớn nhất và phần tử nhỏ nhất của mảng A cùng các chỉ số hàng và cột của 2 phần tử này.
- Trong mảng A có bao nhiêu phần tử bằng phần tử lớn nhất.
- Lập một mảng B mà mỗi cột của A thành một hàng của B.

10. Sắp xếp mảng A[1..m, 1..n] sao cho các phần tử có giá trị tăng dần khi đi theo chiều xoáy trôn ốc từ ngoài vào trong theo chiều kim đồng hồ.

11. Sắp xếp mảng A[1..m, 1..n] sao cho các phần tử có giá trị tăng dần khi đi từ trái sang phải và từ trên xuống dưới.

12. Giả sử n là một số lẻ. Điền các số từ 1 đến n*n vào mảng a[n][n] sao cho tổng các phần tử theo hàng, tổng các phần tử theo cột, tổng các phần tử theo hai đường chéo là bằng nhau. Mảng có tính chất trên gọi là một ma phương.

13. Đọc vào từ bàn phím toạ độ Đề các trong mặt phẳng của 4 điểm A, B, C, D. Kiểm tra xem 4 điểm này có tạo thành một tứ giác lồi hay không ? Nếu có thì tính diện tích của nó.

14. Cho mảng a[m][n]. Phần tử a[i][j] gọi là phần tử yên ngựa nếu nó là phần tử lớn nhất của dòng i và là phần tử nhỏ nhất của cột j, hoặc nó là phần tử nhỏ nhất của hàng i và là phần tử lớn nhất của cột j. In ra tất cả các phần tử yên ngựa của mảng A.

15. Nhập một xâu ký tự từ bàn phím gồm các từ, ví dụ ST = ' Thu do Ha noi'. Lập chương trình để bỏ bớt các dấu trống giữa các từ sao cho các từ chỉ cách nhau ít nhất một dấu trống.

16. Viết chương trình nhập vào từ bàn phím Họ và Tên của một người, sau đó in phần Tên ra màn hình. Ví dụ nhập ' Tran Hung Dao ' thì in ra 'Dao'.

17. Lập trình nhập vào từ bàn phím danh sách học sinh một lớp, sắp xếp lại danh sách theo thứ tự abc của tên, nếu trùng tên thì sắp theo thứ tự abc của họ và tên đệm.

18. Nhập vào một câu kết thúc bằng dấu chấm. In ra câu đó có bao nhiêu từ.

19. Nhập một số nguyên hệ 10, in ra dạng chữ của số. Ví dụ: 3605076, in ra "ba triệu sáu trăm linh năm ngàn không trăm bảy mươi sáu".

20. Viết chương trình đổi một số nguyên dương từ hệ cơ số 10 sang hệ cơ số bất kỳ (cơ số 2, cơ số 8, cơ số 16).

21. Nhập một xâu S từ bàn phím, xâu S gồm n số thực được viết cách nhau bởi dấu phẩy, ví dụ S ='3.2, 8.567, -98.12, 0.1234, -3.45'. Hãy chuyển n số từ xâu vào một mảng số thực A[1..n].

22. Viết chương trình nhập vào từ bàn phím một kí tự và một xâu ký tự. Hãy thông báo ra màn hình kí tự đó xuất hiện bao nhiêu lần trong xâu kí tự và tại những vị trí nào.

23. Viết chương trình nhập vào từ bàn phím hai số nguyên dương M, N. Hãy tìm cách thay các dấu ? trong biểu thức `(((M?M) ?M) ?M) ?M)` bởi các phép toán `+-*/ div` sao cho giá trị của biểu thức nhận được bằng N.

24. Lập trình nhập vào từ bàn phím các tập A, B mà các phần tử của chúng thuộc các ký tự từ 'A' tới 'Z'. Xác định các tập C, D, E, trong đó C = A*B, D = A+B, E = A-B. Đưa ra màn hình lực lượng các tập C, D, E và liệt kê các phần tử của từng tập hợp.

25. Lập trình nhập vào từ bàn phím các tập A, B mà các phần tử của chúng là những số nguyên dương trong phạm vi từ 0 đến 255. Xác định các tập C, D, E, trong đó C = A*B, D = A+B, E = A-B. Đưa ra màn hình lực lượng các tập C, D, E và liệt kê các phần tử của từng tập hợp.

26. Lập trình nhập từ bàn phím một tập W gồm các phần tử là chữ cái trong phạm vi từ C đến R và từ c đến r, khi nhập số 0 thì kết thúc nhập. Với xâu S nhập từ bàn phím hãy cho biết có bao nhiêu kí tự của S thuộc W. Xây dựng tập U gồm các kí tự của S nhưng không thuộc W, in tập U.

27. Lập chương trình nhập từ bàn phím 4 tập A, B, C, D có phần tử là những số nguyên dương. Tìm tập có lực lượng nhỏ nhất. Tìm hai tập có giao lớn nhất trong số giao của từng cặp hai tập khác nhau từng đôi một. Tìm 3 tập có hợp lớn nhất.

28. Nhập từ bàn phím hai tập A và B có phần tử là những ký tự của bảng mã ASCII, dùng phím ESC (mã 27) để kết thúc nhập một tập. Gọi C là tập các ký tự của bảng mã ASCII. Tìm các tập: phần bù của A, phần bù của B, phần bù của A∩B đối với C.

Chương 4

Chương trình con

Trong Turbo Pascal có hai loại chương trình con: hàm và thủ tục. Ưu điểm của việc dùng chương trình con: tránh viết lặp lại một đoạn chương trình nhiều lần trong chương trình, mẫu hoá một đoạn chương trình để cho nhiều người dùng (người dùng chỉ cần biết dữ liệu vào và dữ liệu ra để ở các biến nào mà không cần biết chi tiết thuật toán trong đoạn chương trình), phân một chương trình lớn thành nhiều mô đun độc lập.

1. Hàm và thủ tục

a) Cấu trúc tổng quát

• Cấu trúc của hàm có dạng:

```
Function Tên_hàm(Tham_số_1: kiểu; Tham_số_2: kiểu; . . .) : kiểu;
Var  Khai báo các biến cục bộ ;
Begin  Các lệnh tính toán ;
       Tên_hàm := Biểu thức giá trị ;
End;
```

Lời gọi hàm : Tên_hàm(danh sách các tham số thực sự)

• Cấu trúc của thủ tục:

```
Procedure Tên_thủ_tục(Tham_số_1: kiểu; Tham_số_2: kiểu; . . .;
                      Var Tham_so_3: kiểu; Var Tham_số_4: kiểu) ;
Var  Khai báo các biến cục bộ ;
Begin  Các lệnh tính toán ;
End;
```

Lời gọi thủ tục : Tên_thủ_tục(danh sách các tham số thực sự);

• Giải thích

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

Sự khác nhau cơ bản giữa hàm và thủ tục: hàm cho ra một giá trị thông qua tên hàm và hàm có thể tham gia vào các biểu thức tính toán, còn thủ tục cho ra nhiều giá trị thông qua các tham số biến. Lời gọi thủ tục là một lệnh của Turbo Pascal. Trong thân hàm bao giờ cũng có lệnh gán giá trị cho tên hàm.

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

Trong thủ tục các tham số giá trị thường là các biến để chứa dữ liệu đưa vào thủ tục, các tham số biến là các biến mà kết quả tính toán của thủ tục sẽ chứa vào đó khi ra khỏi thủ tục và ta có thể dùng chúng để tính toán tiếp. Cần hết sức tránh việc nhập dữ liệu và in kết quả trong ruột chương trình con. Trong lập trình người ta cũng hay dùng thủ tục không có tham số, điều đó chỉ cốt giúp cho lập trình sáng sủa dễ đọc.

Trong một chương trình chính (cấp 0, cấp cao nhất) ta có thể khai báo các chương trình con (cấp 1) trước thân chương trình chính, trong mỗi chương trình con cấp 1 ta lại có thể khai báo các chương trình con (cấp 2) trước thân của chương trình con cấp 1 này . . . Các biến khai trong phần Var của chương trình chính được dùng trong toàn bộ chương trình. Các biến khai sau phần Var của chương trình con cấp n sẽ được dùng trong chương trình con này và tất cả các chương trình con cấp thấp hơn n+1, n+2, . . .

Các chương trình con được gọi là ngang cấp nếu chúng cùng trực thuộc hoặc chương trình chính hoặc một chương trình con khác. Khi gọi chương trình con ta phải tuân theo các quy tắc sau:

- Trong thân chương trình (chính hoặc con) có thể gọi tới các chương trình con trực thuộc nó.

- Từ trong một chương trình con (trong thân của chương trình con này hoặc trong thân của các chương trình con bên trong chương trình con này) có thể gọi tới một chương trình con ngang cấp và đã xây dựng trước nó.

- Trong thân một chương trình (chính hay con) không cho phép gọi tới các chương trình con của nó nhưng không thuộc nó.

Vì những quy tắc trên ta phải luôn chú ý tới thứ tự các chương trình con. Song Turbo Pascal cung cấp cho ta từ khoá Forward để tránh điều này: nếu một thủ tục hay hàm được khai báo trước bằng Forward thì nó có thể được sử dụng trong các chương trình con khác được xây dựng trước nó. Cú pháp có dạng:

```
program ChuongTrinhChinh;
var ...
procedure ThuTuc1(cac tham so hinh thuc); forward;
function  Ham1(cac tham so hinh thuc); forward;

procedure ThuTuc2(cac tham so hinh thuc);
begin  .....  ThuTuc1(cac tham so thuc su);
              S := Ham1(cac tham so thuc su); .....
end;

procedure ThuTuc1(cac tham so hinh thuc);
begin   Xay dung than thu tuc .... end;
function Ham1(cac tham so hinh thuc) : kieu;
begin Xay dung than ham .... end;

Begin  Than chuong trinh chinh ..... End.
```

b) Biến toàn cục, biến cục bộ và cơ chế truyền tham số

Mỗi byte trong bộ nhớ là một ô nhớ. Byte đầu tiên có địa chỉ vật lý là 0, byte thứ hai có địa chỉ là 1, . . . , cứ như vậy ta đánh địa chỉ hết bộ nhớ. Trong lập trình người ta thường dùng địa chỉ logic. Bộ nhớ trong 640 KB của máy tính được chia thành các segment (đoạn) có độ lớn 64 KB, các segment đánh số từ $0000 đến $FFFF (ký tự $ phía trước để chỉ số viết ở hệ 16). Segment $0000 bắt đầu từ byte thứ 0 của bộ nhớ, segment $0001 bắt đầu từ byte thứ 16 của bộ nhớ, các segment bắt đầu từ các địa chỉ cách nhau 16 byte. Như vậy các segment có sự chồng chéo nhau. Các byte trong một segment lại được đánh địa chỉ từ $0000 đến $FFFF, địa chỉ này gọi là địa chỉ offset.

Địa chỉ logic có dạng Segment:Offset, ví dụ SA4FB:$487C. Turbo Pascal cung cấp các hàm sau liên quan tới địa chỉ:

- Hàm **Seg(x)**: word cho địa chỉ segment và hàm **Ofs(x)**: word cho địa chỉ offset của vùng nhớ cấp phát cho đối tượng x (x có thể là biến, hàm hay thủ tục).

- Hàm **Sizeof(x)**: word cho kích thước của x tính theo byte (x có thể là tên biến hay tên kiểu dữ liệu).

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

Biến cục bộ là biến được khai báo ở đầu một chương trình con và các tham số của chương trình con khai báo không có Var. Chúng được cấp phát bộ nhớ khi chương trình con được gọi tới và bị xoá khi máy ra khỏi chương trình con. Biến cục bộ có giá trị trong chương trình con và tất cả các chương trình con khác nằm trong chương trình con này. Nếu tên biến cục bộ của một chương trình con trùng với một tên biến toàn cục thì máy cũng không nhầm lẫn, máy sẽ dùng hai vùng nhớ khác nhau để lưu trữ hai biến, khi ra khỏi chương trình con biến cục bộ bị xoá.

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

- Cấp phát bộ nhớ mới cho các đối khai không có Var và các biến cục bộ. Với các đối số khai có Var của chương trình con thì vẫn dùng vùng nhớ tương ứng của tham số thực sự thuộc chương trình mẹ.

- Truyền giá trị của các tham số thực sự cho các tham số giá trị tương ứng, truyền địa chỉ của các tham số thực sự ứng với tham số biến cho các tham số biến của thủ tục.

- Thực hiện các lệnh trong chương trình con. Trong khi thực hiện chương trình con, các biến cục bộ và các tham số giá trị có thể bị biến đổi nhưng không ảnh hưởng tới các biến bên ngoài. Trái lại mọi thay đổi của tham số biến trong chương trình con sẽ kéo theo sự thay đổi của tham số thực sự tương ứng (vì có sự truyền theo địa chỉ). Do đó khi ra khỏi chương trình con, các biến tham số thực sự ứng với tham số biến vẫn giữ được các giá trị mới nhất do chương trình con tạo ra.

- Thực hiện xong các lệnh của chương trình con, máy giải phóng các đối và các biến cục bộ, trở về nơi gọi chương trình con.

Chương trình **HThang.pas**: lập hàm tính diện tích hình thang.

```
var dai, ngan, cao, s: real;
function HinhThang(a,b,h: real) : real;
begin HinhThang := (a+b)*h/2
end;
begin
write('Vao day dai, day ngan, chieu cao : ');
readln(dai,ngan,cao);
s := HinhThang(dai,ngan,cao);
writeln('Dien tich hinh thang = ',s:1:5); readln;
end.
```

Chương trình **HinhCau.pas**: lập thủ tục để tính diện tích và thể tích hình cầu, thủ tục trả về hai giá trị.

```
uses crt;
var bk, dt, tt : real;
procedure Hinhcau(r: real; var s,v: real);
begin
writeln('Dia chi bien r la ',seg(r),':',ofs(r),', r = ',r:1:5);
writeln('Dia chi bien s la ',seg(s),':',ofs(s));
writeln('Dia chi bien v la ',seg(v),':',ofs(v));
s := 4*pi*r*r; v := 4*pi*r*r*r/3
end;
begin clrscr;
write('Vao ban kinh : '); readln(bk);
writeln('Dia chi bien bk la ',seg(bk),':',ofs(bk));
writeln('Do lon vung nho chua bk = ',sizeof(bk));
writeln('Dia chi bien dt la ',seg(dt),':',ofs(dt));
writeln('Dia chi bien tt la ',seg(tt),':',ofs(tt));
HinhCau(bk,dt,tt);
writeln('Dien tich = ',dt:1:5,' The tich = ',tt:1:5);
writeln('Dia chi thu tuc HinhCau la ',seg(HinhCau),':',ofs(HinhCau));
readln;
end.
```

Khi chạy chương trình biến bk và r được cấp hai vùng nhớ khác nhau, biến dt và s được cấp cùng vùng nhớ, biến tt và v cũng được cấp cùng vùng nhớ. Kích thước của tất cả các biến đều là 6 byte.

Chương trình **ChuNhat.pas**: tính diện tích và chu vi hình chữ nhật bằng ba thủ tục không có tham số (chỉ cốt làm cho chương trình mạch lạc).

```
var a,b,dt,cv: real;
procedure NhapDuLieu;
begin write('Chieu dai va chieu rong : '); readln(a,b);
end;
procedure TinhToan;
begin dt:= a*b; cv:= 2*(a+b);
end;
procedure InKetQua;
begin writeln('Dien tich = ',dt:1:5,' Chu vi = ',cv:1:5);
end;
BEGIN NhapDuLieu; TinhToan; InKetQua; readln;
END.
```

Chương trình **TimKiem.pas**: cho một mảng a có n phần tử và một số x, tìm trong mảng a xem có phần tử nào trùng với x hay không, hàm TimKiem() cho giá trị 0 nếu không tìm thấy và cho chỉ số của phần tử mảng trùng với x. Đối của hàm là một mảng một chiều.

```
const nmax = 50;
type mang = array[1..nmax] of integer;
var a: mang; i,j,n,x: integer;

function TimKiem(n:integer; a:mang; x: integer): integer;
var i,k: integer;
begin k := 0;
for i:=1 to n do if a[i]=x then begin k:=i; break end;
TimKiem := k;
end;
```

```
begin
write('Vao so phan tu cua mang n = '); readln(n);
for i:=1 to n do begin write('a[',i,'] = '); readln(a[i]) end;
while true do
 begin write('Vao gia tri can tim x = '); readln(x);
   if x=0 then break; j := TimKiem(n,a,x);
   if j=0 then writeln('Khong thay phan tu ',x,' trong mang')
   else writeln('Phan tu thu ',j,' cua mang trung voi ',x);
 end;
end.
```

Chương trình **NhanMt.pas**: lập một thủ tục nhân hai ma trận a[1..m,1..n] và b[1..n,1..p], ma trận kết quả là c[1..m,1..p]. Cả ba đối của thủ tục đều là các mảng hai chiều.

```
const mmax=30; nmax=30; pmax=30;
type
  matran1= array[1..mmax,1..nmax] of real;
  matran2= array[1..nmax,1..pmax] of real;
  matran3= array[1..mmax,1..pmax] of real;
var a1: matran1; b1: matran2; c1: matran3; m1,n1,p1,i,j: integer;

procedure nhmatran(m,n,p:integer;a:matran1;b:matran2;var c:matran3);
var k,i,j: integer;
begin  for i:=1 to m do for j:=1 to p do
         begin  c[i,j]:=0.0;
                for k:=1 to n do c[i,j]:= c[i,j]+a[i,k]*b[k,j];
         end;
end;

begin
write('Vao m,n,p = '); read(m1,n1,p1);
for i:=1 to m1 do  for j:=1 to n1 do
    begin write('a[',i,',',j,'] = '); readln(a1[i,j]) end;
for i:=1 to n1 do  for j:=1 to p1 do
    begin write('b[',i,',',j,'] = '); readln(b1[i,j]) end;
nhmatran(m1,n1,p1,a1,b1,c1);
for i:=1 to m1 do
 begin writeln('Dong ',i,' cua C: ');
       for j:=1 to p1 do
          begin  write(c1[i,j]:10:3);
                 if j mod 5 =0 then writeln;
          end;  writeln;
 end;
readln
end.
```

2. Đệ quy

Hàm (hay thủ tục) đệ quy là hàm mà trong thân hàm có lời gọi tới chính nó. Khi hàm gọi đệ quy tới chính nó thì mỗi lần gọi máy sẽ tạo ra một tập các biến cục bộ mới hoàn toàn độc lập với tập các biến cục bộ được tạo trong các lần gọi trước. Có bao nhiêu lần gọi tới hàm thì cũng có bấy nhiêu lần thoát ra khỏi hàm và cứ mỗi lần thoát ra khỏi hàm thì một tập các biến cục bộ bị xoá. Sự tương ứng giữa các lần gọi tới hàm

và các lần ra khỏi hàm được thực hiện theo thứ tự ngược lại: lần ra đầu tiên ứng với lần vào cuối cùng, lần ra cuối cùng ứng với lần đầu tiên gọi hàm.

Chương trình **GiamDem.pas** minh hoạ quá trình diễn ra khi một thủ tục đệ quy được gọi: đầu tiên cho k = 8, gọi thủ tục đệ quy GiamDem, trong thân thủ tục giảm biến dem đi 1, nếu nó còn dương thì lại gọi thủ tục. Ta có thể tưởng tượng thủ tục GiamDem làm việc như ta sao chép 8 lần thân thủ tục GiamDem vào chỗ gọi đệ quy trong thân thủ tục.

```
uses crt;
var k: integer;
procedure Giamdem( dem: integer);
begin
writeln('Gia tri cua bien DEM khi vao thu tuc = ',dem);
dec(dem);
if dem>0 then GiamDem(dem);
writeln('Truoc khi ra thu tuc bien DEM = ',dem);
end;
begin clrscr;
k := 8; GiamDem(8); readln;
end.
```

Chương trình **GiaiThua.pas**: tính giai thừa của số nguyên k nhập từ bàn phím.

```
var k : integer;
function GiaiThua(n: longint): longint;
begin  if (n=0) then GiaiThua:=1 else GiaiThua := n* GiaiThua(n-1);
end;
begin write('Vao k = '); readln(k);
writeln(k,' giai thua la ',GiaiThua(k));
readln;
end.
```

Bài tập

1. Viết hàm cho ước số chung lớn nhất của hai số nguyên dương, hàm cho bội số chung nhỏ nhất của hai số nguyên dương. Sử dụng 2 hàm này để tìm ước số chung lớn nhất và bội số chung nhỏ nhất của N số nguyên dương nhập vào từ bàn phím.

2. Hãy lập các hàm sau và với mỗi hàm lập một chương trình để sử dụng hàm:

- Hàm NguyenTo(x): xác định số x có phải là số nguyên tố hay không. Ví dụ NguyenTo(13) = True, NguyenTo(10) = False.

- Hàm DaoSo(x): cho số viết đảo của số tự nhiên x. Ví dụ: DaoSo(3852) = 2538.

- Hàm DoiXung(x): kiểm tra số tự nhiên x có phải là đối xứng không, tức là các chữ số cách đều đầu và cuối của x có giống nhau không. Ví dụ DoiXung(24642) = True.

- Hàm SapXep(a,n): kiểm tra mảng số thực a[1..n] đã được sắp xếp tăng hay giảm chưa. Hàm cho giá trị 0 nếu mảng chưa sắp xếp, cho giá trị 1 nếu xếp tăng, cho giá trị -1 nếu xếp giảm.

- Hàm TrungBinh(a, n, n1, n2): tính giá trị trung bình của các phần tử liên tiếp từ a[n1] đến a[n2] thuộc dãy số a[1], a[2], . . ., a[n].

3. Dùng hàm GiaiThua(n) để tính tổng

1! + 2! + 3! + 4! + ... + N! (với N <= 12).

4. Lập hàm tính tiền lương TienLuong(SoNgayCong: integer, LuongNgay: real), nếu số ngày công lớn hơn 26 thì Tiền lương = (26 + (Số ngày công -26) * 2) * Lương ngày, nếu số ngày công nhỏ hơn hay bằng 26 thì Tiền lương = Số ngày công * Lương ngày. Lập chương trình chính làm nhiệm vụ: nhập học tên công nhân, tiền lương một ngày, số ngày công trong tháng, in ra các thông tin đã nhập và tiền lương.

5. Số lượng sản xuất của N mặt hàng mà doanh nghiệp sản xuất được cho bởi mảng s = (s1, s2, ..., sn). Giá của N mặt hàng cho bởi mảng g = (g1, g2, ..., gn). Lập một hàm tính tổng giá trị các mặt hàng Giatri(s, g, n) theo công thức: Giatri = s1 * g1 + s2 * g2 + ... + sn * gn.

6. Đa thức cấp N được tính theo công thức

$$F(X) = A_n X^n + A_{n-1} X^{n-1} + ... + A_1 X^1 + A_0$$

Lập hàm DaThuc(a, n, x) để tính giá trị của đa thức.

7. Giả sử hàm F(x) có một nghiệm trong khoảng [a, b] và thoả mãn điều kiện F(a) * F(b) < 0. Để tìm xấp xỉ nghiệm x ta tiến hành chia đôi liên tiếp đoạn [a, b], mỗi lần chia chỉ giữ lại nửa nào có hàm F(x) trái dấu nhau tại hai đầu mút. Khi nào giá trị tuyệt đối của hàm F(x) tại điểm chia xấp xỉ 0 hoặc đoạn chia đủ nhỏ ta sẽ lấy điểm chia làm nghiệm xấp xỉ của phương trình F(x) = 0.

Hãy lập trình để tìm nghiệm của hàm $F(x) = x^2 - 2$ trong khoảng [0, 10] (nghiệm là căn bậc hai của 2). Sau đó sửa chương trình để tìm nghiệm của hàm F(x) = cos(x) với x thuộc khoảng [0, 1].

8. Viết một thủ tục để phân tích số N nguyên dương thành tích các số nguyên tố.

9. Lập thủ tục SapXep(n, s) để sắp xếp mảng s có n phần tử tăng dần. Nhập từ bàn phím 2 mảng số thực: a[1..m], b[1..k]. Tạo một mảng c[m+k] gồm tất cả các phần tử của a và b. Dùng thủ tục SapXep để sắp xếp cả 3 mảng a, b, c theo chiều tăng dần, in 3 mảng đã sắp xếp ra màn hình.

10. Dùng hàm đệ quy để giải bài toán Tháp Hà nội. Cho 3 cọc c1, c2, c3. Tại cọc c1 có xếp n cái đĩa kích thước khác nhau, đĩa nhỏ ở trên, đĩa lớn ở dưới. Hãy tìm cách chuyển chồng đĩa từ cọc c1 sang xếp ở cọc c2 (vẫn đảm bảo đĩa nhỏ ở trên, đĩa lớn ở dưới) bằng cách sử dụng cọc trung gian c3 và tuân theo 3 nguyên tắc sau: mỗi lần chỉ được dịch chuyển một đĩa, mỗi đĩa có thể chuyển từ cọc này sang cọc khác bất kỳ, không được để đĩa lớn lên trên đĩa nhỏ.

11. Ký hiệu các phân số là u/v, u1/u2, v1/v2, w1/w2, trong đó tử số và mẫu số đều là các số nguyên và có thể âm. Lập các thủ tục Nhap(u, v) để nhập một phân số, InPhanSo(u, v) để in ra màn hình một phân số, ToiGian(u, v) để tối giản một phân số, Cong(u1, u2, v1, v2, w1, w2) để cộng hai phân số u1/u2 và v1/v2 được phân số w1/w2, Tru(u1, u2, v1, v2, w1, w2) để trừ hai phân số, Nhan(u1, u2, v1, v2, w1, w2) để nhân hai phân số, Chia(u1, u2, v1, v2, w1, w2) để chia hai phân số. Yêu cầu phân số là kết quả của một phép tính phải được tối giản.

Lập trình nhập vào từ bàn phím hai phân số u1/u2 và v1/v2. Tính tổng, hiệu, tích và thương của hai phân số này, in kết quả ra màn hình.

Chương 5

Dữ liệu kiểu bản ghi

Chương này trình bày cách khai báo và dùng dữ liệu kiểu bản ghi, mảng các bản ghi, các phương pháp sắp xếp một mảng các bản ghi, các phương pháp tìm kiếm trên mảng các bản ghi.

1. Khai báo dữ liệu kiểu bản ghi

Dữ liệu kiểu bản ghi bao gồm nhiều thành phần dữ liệu có kiểu khác nhau. Trong thực tế quản lý ta thường gặp dữ liệu kiểu bản ghi, ví dụ dữ liệu về một sinh viên gồm có các thành phần: họ và tên (kiểu xâu ký tự), năm sinh (kiểu số nguyên), địa chỉ, điểm thi môn toán (kiểu số thực), điểm môn tin, giới tính (kiểu boolean). Mỗi thành phần của một bản ghi gọi là một trường (field).

Chương trình **BanGhi.pas** minh hoạ cách khai báo và dùng dữ liệu kiểu bản ghi.

```
1   uses crt;
2   type sinhvien = record
3                       hoten : string[25];
4                       tuoi : integer;
5                       diemtb : real;
6                     end;
7        date = record
8                   ngay : 1..31;
9                   thang : 1..12;
10                  nam : 1920..2010;
11              end;
12       nhanvien = record
13                      hoten : string[25];
14                      ngaysinh : date;
15                      luong, thuong : real;
16                    end;
17  var sv1, sv2, sv3: sinhvien;
18      nv1, nv2, nv3 : nhanvien; d : date;
19  begin clrscr;
20  sv1.hoten:='Nguyen Van Thanh'; sv1.tuoi:=19; sv1.diemtb:=8.92;
21  sv2:=sv1; sv2.hoten:='Le Binh';
22  write('Ho ten sinh vien 3 : '); readln(sv3.hoten);
23  write('Tuoi va diem trung binh : '); readln(sv3.tuoi,sv3.diemtb);
24  writeln('Sv1: ',sv1.hoten:25,' ',sv1.tuoi:2,' ',sv1.diemtb:5:2);
25  writeln('Sv2: ',sv2.hoten:25,' ',sv2.tuoi:2,' ',sv2.diemtb:5:2);
26  writeln('Sv3: ',sv3.hoten:25,' ',sv3.tuoi:2,' ',sv3.diemtb:5:2);
27  with nv1 do
28    begin write('Ho ten nhan vien 1 : '); readln(hoten);
29          write('Ngay sinh : ');
30          readln(ngaysinh.ngay,ngaysinh.thang,ngaysinh.nam);
31          write('Luong va thuong : '); readln(luong,thuong);
32    end;
33  with nv2, ngaysinh do
34    begin write('Ho ten nhan vien 2 : '); readln(hoten);
```

```
35          write('Ngay sinh : '); readln(ngay,thang,nam);
36          write('Luong va thuong : '); readln(luong,thuong);
37    end;
38  with d do begin ngay:=25; thang:= 12; nam:=1981 end;
39  nv3:=nv1; nv3.hoten:='Tran Minh';
40  nv3.ngaysinh:=d;
41  writeln('Nv1: ',nv1.hoten:25,' ',nv1.ngaysinh.ngay:2,'/',
42           nv1.ngaysinh.thang:2,'/',nv1.ngaysinh.nam:4,' ',
43           nv1.luong:7:0,' ',nv1.thuong:7:0);
44  with nv2,ngaysinh do
45    writeln('Nv2: ',hoten:25,' ',ngay:2,'/',thang:2,'/',
46                  nam:4,' ',luong:7:0,' ',thuong:7:0);
47  with nv3 do
48     writeln('Nv3: ',hoten:25,' ',ngaysinh.ngay:2,'/',
49       ngaysinh.thang:2,'/',ngaysinh.nam:4,' ',luong:7:0,' ',thuong:7:0);
50  writeln(sizeof(sinhvien),' ',sizeof(date),' ',sizeof(nhanvien));
51  readln;
52  end.
```

Định nghĩa dữ liệu kiểu bản ghi: tên các thành phần dữ liệu kèm theo kiểu đặt giữa hai từ khoá record và end. Dòng 2-6 định nghĩa một kiểu bản ghi có tên là sinhvien với 3 thành phần dữ liệu: hoten kiểu xâu ký tự, tuoi kiểu số nguyên, diemtb kiểu số thực, độ lớn của kiểu sinhvien là 32 byte. Dòng 17 khai báo 3 biến kiểu sinhvien. Dòng 7-11 định nghĩa kiểu bản ghi date có 3 thành phần: ngày, tháng, năm. Dòng 12-16 minh hoạ định nghĩa các kiểu bản ghi lồng nhau, kiểu bản ghi nhanvien có 4 thành phần: họ tên, ngày sinh (có kiểu date đã định nghĩa), lương, thưởng. Dòng 18 định nghĩa 3 biến kiểu nhanvien và một biến kiểu date.

Để thâm nhập vào các trường của một bản ghi ta dùng cách viết: tên biến bản ghi, dấu chấm và tên trường. Dòng 20: gán dữ liệu cho từng trường của biến sv1. Dòng 22-23: nhập dữ liệu cho từng trường của biến sv3 từ bàn phím. Không được dùng lệnh readln và writeln để vào ra một biến kiểu bản ghi mà chỉ dùng các lệnh này đối với các trường đơn (xem dòng 22-26). Dòng 41-43 cho cách truy nhập vào các trường của kiểu bản ghi lồng nhau: tên biến bản ghi • tên trường có kiểu bản ghi • tên trường.

Câu lệnh with. Khi cần thâm nhập nhiều thành phần của một biến kiểu bản ghi ta có thể dùng câu lệnh with để chương trình được gọn hơn. Cú pháp lệnh:

with <tên biến kiểu bản ghi> do <câu lệnh>

Dòng 27-37, dòng 44-49 cho ta hai cách thâm nhập vào các trường của biến bản ghi lồng nhau.

Các phép toán. Các biến bản ghi có cùng kiểu có thể gán cho nhau, các trường của chúng có cùng kiểu cũng thể gán cho nhau: dòng 21, dòng 39, dòng 40. Muốn so sánh hai biến bản ghi có giống nhau hay khác nhau ta phải so sánh tất cả các trường của chúng.

2. Mảng các bản ghi

Mảng các bản ghi là một mảng mà mỗi phần tử của nó là một bản ghi. Để truy nhập tới phần tử thứ i của mảng ta dùng cách viết a[i], a[i] là một biến kiểu bản ghi. Để truy nhập tới các trường của phần tử thứ i của mảng ta dùng cú pháp: **a[i]•tên trường**

hoặc **with a[i] do lệnh**.

Chương trình **MangBG.pas**: lập một báo cáo bán hàng trong ngày của một cửa hàng bán đồ điện lạnh (tí vi, máy điều hoà, tủ lạnh . . .). Thông tin nhập vào gồm có: số hoá đơn, tên mặt hàng, số lượng, giá. Yêu cầu: số lượng hoá đơn cần nhập không quá 20, mã số hoá đơn nhập vào không được trùng với các mã số hoá đơn đã nhập, 1 <= số lượng <= 20, giá > 0, nếu dữ liệu nhập sai máy sẽ thông báo lỗi trên dòng 1 màn hình, các dữ liệu nhập bố trí theo cột trên màn hình. Báo cáo in ra có 5 cột: số hoá đơn, tên hàng, số lượng, giá, thành tiền. Cuối báo cáo có in tổng số tiền thu được trong ngày.

```
uses crt;
const nmax = 100;
type HangHoa = record
                  SoHD: integer;
                  TenHang: string[25];
                  SoLuong: integer;
                  Gia, Tien: real;
               end;
     DanhsachHH = array[1..nmax] of HangHoa;
     xau40 = string[40];
var a: DanhSachHH; n: integer;

procedure TBLoi(s: xau40);
begin gotoxy(40,1); write(s); readkey;
gotoxy(40,1); clreol;
end;

procedure Nhap;
var i,j,x,y: integer; trung: boolean;
begin clrscr; gotoxy(1,2);
write('Vao so luong ban ghi can nhap n = ');
x := wherex; y := wherey;
repeat gotoxy(x,y); clreol; readln(n);
       if (n<1) or (n>20) then TBLoi('Nhap sai n');
until (n>=1) and (n<=20);
gotoxy(1,3); write('SOHD');
gotoxy(6,3); write('TEN MANG');
gotoxy(32,3); write('SO LUONG');
gotoxy(42,3); write('GIA');

for i:=1 to n do
begin
repeat gotoxy(1,3+i); clreol; readln(a[i].SoHD);
       trung := false;
       for j:=1 to i-1 do if a[i].SoHD=a[j].Sohd then
                        begin trung := true; break end;
       if trung then TBLoi('So hoa don bi trung');
until (not trung);
gotoxy(6,3+i); readln(a[i].TenHang);

repeat gotoxy(32,3+i); clreol; readln(a[i].SoLuong);
       if (a[i].SoLuong<1) or (a[i].SoLuong>20) then
              TBLoi('So luong hang khong hop le');
until (a[i].SoLuong>=1) and (a[i].SoLuong<=20);

repeat gotoxy(42,3+i); clreol; readln(a[i].Gia);
       if a[i].Gia<=0 then TBLoi('Gia khong hop le');
```

```
until a[i].Gia > 0;
end;
end;

procedure InKetQua;
var i: integer; tong,t: real;
begin
clrscr; gotoxy(10,2); write('BAO CAO BAN HANG');
gotoxy(1,3); write('SOHD');
gotoxy(6,3); write('TEN HANG');
gotoxy(32,3); write('SO LUONG');
gotoxy(42,3); write('GIA');
gotoxy(58,3); write('TIEN'); tong:= 0;

for i:=1 to n do
  begin
     gotoxy(1,3+i); write(a[i].SoHD);
     gotoxy(6,3+i); write(a[i].TenHang);
     gotoxy(32,3+i); write(a[i].SoLuong:2);
     gotoxy(42,3+i); write(a[i].Gia:1:2);
     t:= a[i].SoLuong*a[i].Gia; tong := tong+t;
     gotoxy(58,3+i); write(t:1:2);
  end;
gotoxy(1,3+n+1); writeln('Tong so tien ban duoc : ',tong:1:2);
readln;
end;
begin  Nhap; InKetQua;
end.
```

Trong chương trình sử dụng các hàm và thủ tục sau của **Unit Crt**: thủ tục **gotoxy(x,y)** chuyển con trỏ tới cột x dòng y, thủ tục **clreol** xoá từ vị trí con trỏ tới cuối dòng, hàm **readkey** để nhận một ký tự từ bàn phím, hàm **wherex** cho toạ độ x của con trỏ, hàm **wherey** cho toạ độ y của con trỏ

3. Các thuật toán sắp xếp

Giả sử ta có một mảng các bản ghi a[1..n], mỗi bản ghi có 3 thành phần : mã nhân viên, họ tên nhân viên, lương chính. Vấn đề đặt ra là cần sắp xếp mảng **a** theo Mã nhân viên tăng dần, khi đó trường MaNV gọi là khoá sắp xếp. Dưới đây trình bày 5 phương pháp sắp xếp thông dụng:

- Phương pháp xen trực tiếp (straight insertion): với mỗi i = 2, 3, . . ., n tìm vị trí để chèn a[i] vào đúng vị trí của nó theo thứ tự tăng dần trong dãy a[1], a[2], . . . , a[i-1].

- Phương pháp xen nhị phân (binary insertion): tương tự phương pháp xen trực tiếp, nhưng khi tìm vị trí thích hợp cho phần tử a[i] ta không sử dụng cách tìm tuần tự mà sử dụng cách tìm nhị phân (chia đôi đoạn a[1], a[2], . . ., a[i-1] để tìm).

- Phương pháp chọn trực tiếp (straight selection): với mỗi phần tử thứ i (i =1, 2,. . ., n-1), mỗi lần có phần tử từ i+1 đến n nhỏ hơn phần tử thứ i thì đổi chỗ hai phần tử cho nhau (kết quả phần tử thứ i là nhỏ nhất trong số các phần tử từ i tới n).

- Phương pháp nổi bọt (bubble sort): sau vòng lặp i =1 phần tử có khoá lớn nhất chìm xuống vị trí a[n] và một số phần tử nổi lên một mức, sau vòng lặp i = 2 phần tử có

khoá lớn thứ hai chìm xuống vị trí a[n-1], . . .

- **Phương pháp sắp xếp nhanh** (quick sort). Bước đầu chọn phần tử đầu tiên a[1] của mảng cần sắp xếp a[1..n] làm mốc, sau đó phân hoạch mảng thành hai phần bằng cách chuyển tất cả các thành phần có khoá lớn hơn khoá của mốc sang bên phải mốc, còn tất cả các thành phần có khoá nhỏ hơn hoặc bằng khoá của mốc được chuyển sang bên trái mốc. Kết quả của phân hoạch, mốc đứng ở vị trí thứ k, các phần tử của mảng con a[1.. k-1] có khoá nhỏ hơn hay bằng khoá của mốc, các phần tử của mảng con a[k+1..n] có khoá lớn hơn khoá của mốc. Sau đó hai mảng con được sắp xếp độc lập bằng cách gọi đệ qui thuật toán trên.

Chương trình **SapXep.pas** cho phép lựa chọn một trong 5 phương pháp sắp xếp một mảng bản ghi. Mỗi phương pháp được lập dưới dạng thủ tục nên rất tiện dùng cho các bài toán khác.

```
uses crt;
const nmax=100;
type nhanvien=record
        manv: string[6];
        hoten: string[23];
        luong: real;
        end;
    dsnhanvien = array[0..nmax] of nhanvien;
var a: dsnhanvien; i,n,chon: integer;

procedure StraightInsertion(n:integer; var a:dsnhanvien);
var i,j: integer; x: nhanvien;
begin
for i:=2 to n do
  begin x:=a[i]; a[0]:=x; j:=i-1;
     while x.manv < a[j].manv do begin a[j+1]:= a[j]; j:= j-1 end;
     a[j+1]:= x;
  end;
end;

procedure BinaryInsertion(n:integer; var a:dsnhanvien);
var i,j,k,q,p: integer; x: nhanvien;
begin
for i:=2 to n do
  begin x:= a[i]; k:=1; p:= i-1;
     while k<= p do begin q:= (k+p) div 2;
                    if x.manv < a[q].manv then p:=q-1 else k:=q+1;
                    end;
     for j:=i-1 downto k do a[j+1]:= a[j];     a[k]:= x;
  end;
end;

procedure StraightSelection(n:integer; var a:dsnhanvien);
var i,j: integer; x: nhanvien;
begin
for i:=1 to n-1 do for j:=i+1 to n do if a[j].manv < a[i].manv then
                  begin x:= a[i]; a[i]:= a[j]; a[j]:=x end;
end;

procedure BubleSort(n:integer; var a:dsnhanvien);
var i,j: integer; x: nhanvien;
```

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

procedure QuickSort(n:integer; var a:dsnhanvien);
procedure QSort(left,right: integer);
var i,j: integer; tg: nhanvien;
begin  if left < right then begin i:=left+1; j:= right end;
repeat  while (a[i].manv <= a[left].manv) and (i<=right) do inc(i);
        while a[j].manv > a[left].manv do dec(j);
        if i < j then begin tg:= a[i]; a[i]:= a[j]; a[j]:= tg end;
until i>=j;
tg := a[left]; a[left]:= a[j];a[j]:= tg;
if left < j-1 then QSort(left,j-1);
if j+1 < right then QSort(j+1,right);
end;
begin  QSort(1,n);
end;

begin clrscr; write('Vao so phan tu cua danh sach: '); readln(n);
for i:=1 to n do with a[i] do
  begin  write('Ma nhan vien: '); readln(manv);
         write('Ho ten : '); readln(hoten);
         write('Luong chinh : '); readln(luong);
  end; clrscr;

writeln('1. Phuong phap xen truc tiep - Straight Insertion');
writeln('2. Phuong phap xen nhi phan - Binary Insertion');
writeln('3. Phuong phap chon truc tiep - Straight Selection');
writeln('4. Phuong phap noi bot - Bubble Sort');
writeln('5. Phuong phap sap xep nhanh - Quick Sort');
write('Hay lua chon phuong phap sap xep : '); readln(chon);
case chon of
1: StraightInsertion(n,a);
2: BinaryInsertion(n,a);
3: StraightSelection(n,a);
4: BubleSort(n,a);
5: QuickSort(n,a);
end;

writeln; writeln('Ket qua sap xep: ');
for i:=1 to n do with a[i] do writeln(manv:8,hoten:26,luong:10:0);
readln;
end.
```

4. Các thuật toán tìm kiếm

Giả sử ta có một mảng bản ghi **a** gồm n phần tử. Cho trước một giá trị **t**, hỏi trong mảng **a** có phần tử nào mà giá trị trường khoá bằng với **t** hay không. Có hai thuật toán tìm kiếm thường dùng: tìm kiếm tuần tự trên mảng **a** chưa được sắp xếp, tìm kiếm nhị phân trên mảng **a** đã được sắp xếp theo trường khoá.

- **Tìm kiếm tuần tự trên mảng a chưa sắp xếp**: ta xét lần lượt từ phần tử thứ nhất đến phần tử thứ n của a, nếu tồn tại a[k].key bằng t thì k là phần tử cần tìm, nếu

không có phần tử nào như vậy thì ta kết luận không có phần tử nào mà trường khoá bằng t.

- Tìm kiếm nhị phân trên mảng a đã được sắp xếp tăng dần. Đầu tiên ta xét phần tử ở giữa của mảng a là a[mid], nếu a[mid].key = t thì a[mid] là phần tử cần tìm và thuật toán dừng lại, nếu t < a[mid].key thì phần tử cần tìm nằm ở nửa đầu của mảng a và ta lại tiếp tục chia đôi nửa đầu, nếu a[mid].key < t thì phần tử cần tìm nằm ở nửa cuối của mảng a và ta lại tiếp tục chia đôi nửa cuối.

Chương trình **TimKiem.pas**: nhập từ bàn phím mảng bản ghi **a**[1..n], mỗi bản ghi có 3 trường: mã số tài khoản, họ tên chủ tài khoản, số dư của tài khoản trong ngân hàng. Nếu sử dụng tìm kiếm nhị phân thì ngay từ lúc nhập mảng **a** đã phải sắp xếp tăng dần theo trường mã số tài khoản, ví dụ nhập các mã số tài khoản 1542, 1678, 2133, 4532, 6678, 7982 . . . (nếu lúc nhập mảng **a** chưa được sắp xếp mà ta lại muốn dùng phương pháp tìm kiếm nhị phân thì trước đó ta phải gọi một thủ tục sắp xếp mảng đã trình bày trong Mục 3). Sau khi chọn chức năng của menu ta cần nhập vào mã số tài khoản cần tìm kiếm, chương trình in ra các thông tin về tài khoản vừa tìm thấy. Thông tin vào của các thủ tục tìm kiếm: mảng **a**, số phần tử của mảng **n**, mã số tài khoản cần tìm **tk**; thông tin ra: số nguyên **k**, k=0 là không tìm thấy, nếu tìm thấy thì a[k] là phần tử có mã tài khoản trùng với tk.

```
uses crt;
const nmax = 100;
type taikhoan = record
                  sotk : integer;
                  chutk : string[23];
                  sodu : real;
                end;
    dstaikhoan = array[1..nmax] of taikhoan;
var a: dstaikhoan; i,n,chon,tk,k: integer;

procedure TkTuanTu(n:integer;a:dstaikhoan;tk:integer;var k:integer);
var i: integer;
begin k := 0;
for i:=1 to n do if a[i].sotk = tk then begin k := i; break end;
end;

procedure TkNhiPhan(n:integer;a:dstaikhoan;tk:integer;var k:integer);
var mid, low, high: integer;
begin low:=1; high:=n; k:=0;
while low<=high do
   begin mid := (low + high) div 2;
         if tk < a[mid].sotk then high := mid-1
         else if tk > a[mid].sotk then low := mid+1
              else begin k := mid; break end;
   end;
end;

begin clrscr; write('Vao so phan tu cua danh sach: '); readln(n);
for i:=1 to n do with a[i] do
  begin  write('Ma so tai khoan: '); readln(sotk);
         write('Ho ten chu tai khoan : '); readln(chutk);
         write('So du tai khoan : '); readln(sodu);
  end;
```

```
while true do begin
writeln('1. Tim kiem tuan tu (tep khong can sap xep)');
writeln('2. Tim kiem nhi phan tren tep da sap xep');
writeln('3. Ket thuc chuong trinh');
write('Chon chuc nang : '); readln(chon);
if (chon<>1) and (chon<>2) then break;
write('Vao ma tai khoan can tim : '); readln(tk);
if chon=1 then TkTuanTu(n, a, tk, k)
          else TkNhiPhan(n, a, tk, k);
if k=0 then writeln('Khong tim thay tai khoan nay')
       else writeln(a[k].sotk,' ',a[k].chutk,' ',a[k].sodu:1:5);
readln;
              end;
end.
```

Bài tập

1. Để quản lý nhân viên trong cơ quan ta khai báo một kiểu bản ghi gồm các trường: mã nhân viên (ma_nv: string[10]), họ tên nhân viên (ho_ten: string[25]), lương chính (luong: real), phụ cấp (phu_cap: real), bảo hiểm xã hội (xa_hoi: real, bằng 5% lương chính), bảo hiểm y tế (y_te: real, bằng 1% lương chính), tổng số tiền được lĩnh trong tháng (tong: real, bằng luong + phu_cap - xa_hoi - y_te).

Lập trình tạo một menu điều khiển chương trình gồm các mục:

- Nhập dữ liệu cho n nhân viên vào một mảng bản ghi, riêng 3 trường xa_hoi, y_te và tong không nhập mà tính toán theo công thức.
- In toàn bộ các phần tử của mảng bản ghi ra màn hình.
- Sắp xếp mảng bản ghi theo trường ma_nv tăng dần, in mảng đã sắp xếp.
- Thoát khỏi chương trình.

2. Để quản lý thí sinh thi vào một trường ta khai báo một kiểu bản ghi gồm các trường: số báo danh (sbd: string[10]), họ và tên thí sinh (hoten: string [25]), ngày sinh (ngaysinh: date, date cũng là kiểu bản ghi có 3 trường: ngày, tháng và năm), điểm thi môn thứ nhất (mon1: real), điểm thi môn thứ hai (mon2: real), điểm thi môn thứ ba (mon3: real), điểm ưu tiên (uutien: real), tổng số điểm (tongso: real, bằng mon1 + mon2 + mon3 + uutien).

Tạo một menu gồm các chức năng:

- Nhập dữ liệu của n thí sinh vào một mảng bản ghi, trường tongso không nhập mà tính theo công thức.
- Sắp xếp danh sách thí sinh theo số báo danh tăng dần, vào từ bàn phím một số báo danh, dùng phương pháp tìm kiếm nhị phân để tìm thí sinh có số báo danh vừa nhập và in các điểm của thí sinh này.
- Nhập từ bàn phím một điểm chuẩn (ví dụ 21.5 điểm), in ra danh sách các thí sinh trúng tuyển theo số báo danh tăng dần (in tất cả các trường), in tổng số thí sinh trúng tuyển. In ra danh sách các thí sinh không trúng tuyển cũng theo số báo danh tăng dần. Thí sinh trúng tuyển là thí sinh có tổng số điểm lớn hơn hay bằng điểm chuẩn và không môn nào bị điểm liệt (<=1).
- In danh sách các thí sinh theo tổng số điểm giảm dần, bảng in ra có 3 cột: số báo danh, họ tên thí sinh, tổng số điểm.
- Kết thúc chương trình.

Chương 6

Dữ liệu kiểu tệp

Khi bài toán cần nhiều dữ liệu hay khi muốn sử dụng dữ liệu nhiều lần thì ta phải tổ chức lưu trữ dữ liệu trên đĩa dưới dạng các tệp, gọi là dữ liệu kiểu tệp. Khi kết thúc chương trình hay khi tắt máy dữ liệu kiểu tệp vẫn còn trên đĩa. Trong Turbo Pascal có ba loại tệp: tệp định kiểu, tệp văn bản và tệp không định kiểu.

1. Tệp định kiểu

Tệp có định kiểu là tệp mà tất cả các phần tử của nó đều thuộc cùng một kiểu dữ liệu nhất định (integer, real, double, char, mảng, bản ghi . . .). Cú pháp khai báo:

Type Tên_kiểu_tệp = File of Kiểu_dữ_liệu ;
Var Biến_tệp : Tên_kiểu_tệp ;

Ví dụ :
```
Type NhanVien = record
                  MaNV : integer ;
                  HoTen : string[25] ;
                  GioiTinh: char;
                  Luong : real ;
                end;
     TepNhanVien = File of NhanVien ;
Var  F : TepNhanVien ;
```

Lệnh trên khai báo biến tệp F là tệp có định kiểu, mỗi phần tử của tệp là một bản ghi có kích thước 34 byte (trường Mã nhân viên chiếm 2 byte, trường Họ tên 26 byte, trường Lương chính 6 byte). Các bản ghi xếp liền nhau trên tệp, phần tử đầu tiên có số hiệu là 0, phần tử tiếp theo có số hiệu là 1, ..., cuối tệp có dấu kết thúc tệp. Khi tệp được mở để làm việc ta luôn luôn có một con trỏ tệp. Lúc mới mở tệp con trỏ tệp trỏ vào phần tử thứ 0 (byte đầu tiên của phần tử này), cứ sau mỗi lần đọc (read) hay ghi (write) con trỏ tệp tự động dịch chuyển tới phần tử kế tiếp (dịch 34 byte). Ta có thể dùng lệnh Seek(f,n) để di chuyển con trỏ tệp tới phần tử thứ n bất kỳ. Có thể xem tệp như một mảng được lưu trữ ở bộ nhớ ngoài, con trỏ tệp tương ứng với biến chỉ số của mảng.

0 Con trỏ tệp ⇓1 2

1234	Le An	850000	4532	Tran Huy	1200000	5677	Vu Ban	2500000	♦

Việc truy nhập tới các phần tử của một tệp được thực hiện qua hai cách: truy nhập tuần tự và truy nhập trực tiếp. Truy nhập tuần tự: việc đọc một phần tử bất kỳ của tệp phải đi qua các phần tử trước đó; muốn thêm một phần tử vào tệp phải đặt con trỏ tệp vào cuối tệp. Truy nhập trực tiếp: ta có thể đặt con trỏ tệp vào một phần tử bất kỳ của tệp.

Biến tệp quản lý vùng đệm mà hệ thống cung cấp khi làm việc với tệp, nó không chứa nội dung dữ liệu như các biến khác, vì thế không thể dùng phép gán cho các biến tệp cũng như truyền theo trị cho các tham số kiểu tệp, tham số kiểu tệp trong các chương trình con phải là tham biến.

Các hàm và thủ tục dùng cho cả ba loại tệp:

• **Thủ tục Assign(var f; TenTep: string)**: gán một tên tệp trên đĩa cho biến tệp f. TenTep là một xâu ký tự chứa tên tệp (có thể có cả đường dẫn). Sau lệnh này biến f sẽ đại diện cho tệp đã đăng ký cho đến khi đăng ký lại. Tham biến f trong các hàm và thủ tục về sau đều được hiểu là biến tệp đã đăng ký với một tên tệp nào đó.

• **Thủ tục Reset(f)**: mở tệp đã đăng ký với biến tệp f, sau đó có thể đọc dữ liệu trên tệp và ghi dữ liệu lên tệp. Nếu tệp chưa có trên đĩa máy sẽ báo lỗi.

• **Thủ tục Rewrite(f)**: mở một tệp hoàn toàn mới để ghi dữ liệu mới. Nếu tệp đã có trên đĩa thì dữ liệu cũ sẽ bị xoá hết.

• **Thủ tục Close(f)**: đóng tệp. Khi không làm việc với tệp cần phải đóng tệp để đảm bảo an toàn dữ liệu.

• **Hàm Eof(f)**: cho giá trị True nếu con trỏ tệp ở vị trí cuối tệp (sau phần tử cuối cùng của tệp), nếu trái lại hàm cho giá trị False. Hàm này thường dùng để kiểm tra đã đọc hết dữ liệu trong tệp chưa.

• **Thủ tục Erase(f)**: xoá tệp đã đăng ký với biến tệp f, không được xóa tệp đang mở. Ví dụ: `assign(f,'hoadon.dat'); erase(f);`

• **Thủ tục Rename(var f; Str: string)**: đổi tên tệp đã đăng ký với biến tệp f thành tên tệp chứa trong xâu Str. Không được đổi tên tệp đang mở và tên mới không được trùng với tên tệp đã có trên đĩa đang làm việc. Ví dụ: `assign(f,'nv.dat'); rename(f,'nhanvien.dat');`

Các hàm và thủ tục dùng cho tệp định kiểu:

• **Thủ tục Read(f, x)**: đọc một phần tử trên tệp f tại vị trí con trỏ tệp và lưu vào biến x, đọc xong con trỏ tệp di chuyển tới phần tử kế tiếp. Biến x phải cùng kiểu với kiểu phần tử của tệp. Thủ tục này có thể đọc cho nhiều biến Read(f, danh sách biến).

• **Thủ tục Write(f, x)**: ghi nội dung của biến x thành một phần tử của tệp f mà con trỏ tệp đang định vị, ghi xong con trỏ tệp dịch chuyển tới phần tử kế tiếp. Biến x phải cùng kiểu với kiểu phần tử của tệp. Thủ tục này có thể ghi nhiều biến lên tệp Write(f, danh sách biến).

• **Thủ tục Seek(var f; k: longint)**: đặt con trỏ tệp vào byte đầu tiên của phần tử có số hiệu là k (số hiệu phần tử đầu tiên là 0)

• **Hàm FileSize(var f): longint** cho số phần tử của tệp f. Lệnh seek(f, filesize(f)) sẽ đưa con trỏ tới cuối tệp, sau đó ta có thể ghi bổ sung vào cuối tệp.

• **Hàm FilePos(f): longint** cho biết vị trí hiện tại của con trỏ tệp đang ở phần tử có số hiệu là bao nhiêu.

• **Thủ tục Truncate(f)** : cắt bỏ các phần tử tính từ vị trí con trỏ tệp cho đến hết tệp, vị trí hiện tại của con trỏ trở thành vị trí cuối tệp.

Chương trình **Nguyen.pas**: tạo một tệp định kiểu Nguyen.dat gồm các số nguyên nhập vào từ bàn phím (chọn chức năng 1), mở tệp định kiểu Nguyen.dat để đọc các số nguyên đưa vào mảng a (chọn chức năng 2).

```
Var i,j,n,x,chon : integer;
    a : array[1..100] of integer;
```

```
    f : file of integer;
begin
write('1. Tao tep so nguyen  2. Doc tep. Chon:  '); readln(chon);
case chon of
1: begin assign(f,'Nguyen.dat'); rewrite(f);
   write('Vao so phan tu n = '); readln(n);
   for i:=1 to n do
    begin write('So ',i,'= '); readln(x); write(f,x); end; close(f);
   end ;
2: begin assign(f,'Nguyen.dat'); reset(f); i:=0;
   while not eof(f) do begin inc(i); read(f,x); a[i]:=x end;
close(f);
   for j:=1 to i do  writeln('a(',j,') = ',a[j]);
   end;
end;
write('An Enter de ket thuc chuong trinh'); readln;
end.
```

Chương trình **QuanLy.pas** dùng để quản lý các nhân viên trong cơ quan với 9 chức năng chính sau:

1. Tạo một tệp định kiểu NVien.dat (còn gọi là tệp cơ sở dữ liệu) mới, mỗi phần tử của tệp là một bản ghi (có kích thước 37 byte) gồm 3 trường: mã nhân viên, họ tên nhân viên, lương chính. Chức năng này được thực hiện bằng cách gọi thủ tục Tao_BoSung() với đối new là True. Dữ liệu các bản ghi nhập vào từ bàn phím, nếu chọn 'K' khi máy hỏi "Tiep tuc nhap C/K ?" thì sẽ kết thúc nhập.

2. Bổ sung vào cuối tệp NVien.dat các bản ghi mới. Chức năng này được thực hiện bằng cách gọi thủ tục Tao_BoSung() với đối new là False.

3. Sắp xếp cơ sở dữ liệu theo trường mã nhân viên tăng dần (dùng thủ tục SapXep). Thuật toán: chuyển các bản ghi từ tệp NVien.dat vào mảng a, dùng thuật toán nổi bọt sắp xếp mảng a theo chiều tăng của trường MaNV, xếp xong ghi mảng a vào tệp NVienSX.dat.

4. Xem tệp cơ sở dữ liệu chưa sắp xếp NVien.dat: dùng thủ tục InCSDL.

5. Xem tệp cơ sở dữ liệu đã được sắp xếp NVienSX.dat: dùng thủ tục InCSDL.

6. Tìm kiếm bản ghi theo mã nhân viên và sửa bản ghi vừa tìm thấy (dùng thủ tục TimSua). Thuật toán: vào từ bàn phím một mã nhân viên, tìm trong tệp NVien.dat bản ghi có mã nhân viên trùng với mã vừa nhập, nếu tìm thấy cho hiện nội dung bản ghi này ra màn hình, máy cho phép chọn trường cần sửa theo số thứ tự (nhập 0 để kết thúc sửa), sửa xong máy ghi đè lên bản ghi cũ của tệp.

7. Tìm kiếm bản ghi theo mã nhân viên và xoá bản ghi vừa tìm thấy (dùng thủ tục TimXoa). Thuật toán: chuyển các bản ghi từ tệp NVien.dat vào mảng a[1..n], vào từ bàn phím mã nhân viên cần xóa, dùng thuật toán tìm kiếm tuần tự trên mảng a để tìm phần tử có mã nhân viên trùng với mã vừa nhập (giả sử là a[k]), ghi toàn bộ mảng trở lại tệp NVien.dat trừ phần tử a[k].

8. Chèn một bản ghi mới (dùng thủ tục ChenBG). Thuật toán: chuyển các bản ghi từ tệp NVien.dat vào mảng a, in nội dung các phần tử của mảng cùng với số thứ tự, nhập vị trí cần chèn bản ghi mới (trước bản ghi thứ mấy của mảng a, ký hiệu là j), vào

nội dung bản ghi mới từ bàn phím, chèn phần tử mới vào vị trí thứ j của mảng a và đẩy các phần tử từ j đến n ra phía sau, ghi toàn bộ mảng a vào tệp NVien.dat.

9. Tìm kiếm bản ghi theo mã nhân viên trên tệp đã sắp xếp NVienSX.dat và in ra màn hình bản ghi vừa tìm thấy (dùng thủ tục TimIn). Thuật toán: chuyển các bản ghi từ tệp NVienSX.dat vào mảng a, vào mã nhân viên cần tìm, dùng thuật toán tìm kiếm nhị phân để tìm phần tử a[k] của mảng a có mã nhân viên trùng với mã vừa nhập, in nội dung bản ghi a[k] ra màn hình.

```
uses crt;
const nmax=100;
type nhanvien=record
                manv: string[6];
                hoten: string[23];
                luong: real;
              end;
    dsnhanvien = array[1..nmax] of nhanvien;
    tepnhanvien = file of nhanvien;
    xau12 = string[12];
var chon: integer;

procedure Tao_BoSung(new: boolean);
var f1: tepnhanvien; tiep: char; X: nhanvien;
begin
if new then begin assign(f1,'NVien.dat'); rewrite(f1);  end
       else begin assign(f1,'NVien.dat'); reset(f1);
                  seek(f1,filesize(f1));  end;       tiep:='c';
while (tiep='C') or (tiep='c') do
  begin  with X do
           begin  write('Ma nhan vien: '); readln(manv);
                  write('Ho ten : '); readln(hoten);
                  write('Luong chinh : '); readln(luong);
           end;
  write(f1,X); write('Tiep tuc nhap C/K ? '); readln(tiep);
end;  close(f1);
end;

procedure SapXep;
var f1,f2: tepnhanvien; a: dsnhanvien; i,j,n: integer; x: nhanvien;
begin  assign(f1,'NVien.dat'); reset(f1);
n:=filesize(f1); for i:=1 to n do  read(f1,a[i]); close(f1);
for i:=1 to n-1 do for j:=1 to n-i do if a[j].manv>a[j+1].manv then
                          begin x:=a[j]; a[j]:=a[j+1]; a[j+1]:=x end;
assign(f2,'NVienSX.dat'); rewrite(f2);
for i:=1 to n do write(f2,a[i]); close(f2);
write('Sap xep xong .....'); readln;
end;

procedure InCSDL(tentep: xau12);
var f1: tepnhanvien; x: nhanvien; i: integer;
begin assign(f1,tentep); reset(f1);
for i:=1 to filesize(f1) do
  begin read(f1,X);
        writeln(i,'. ',X.manv:6,' ',X.hoten:25,' ',X.luong:8:0);
  end;    close(f1); write('An Enter .....'); readln;
end;
```

```
procedure TimSua;
var f1: tepnhanvien; k,truong: integer; TimThay: boolean;
    x:nhanvien; ma: string[6];
begin assign(f1,'NVien.dat'); reset(f1);
write('Ban can tim nhan vien co ma : '); readln(ma);
k:=0; TimThay:=false;
while not eof(f1) do
  begin read(f1,X);if X.manv=ma then begin Timthay:=True; break end;
        inc(k);  end;
if TimThay then begin
  while true do begin
  writeln('1. Ma nhan vien: ',X.manv);
  writeln('2. Ho va ten : ',X.hoten);
  writeln('3. Luong : ',X.luong:6:0);
  write('Chon truong can sua 1..3, chon 0 neu ket thuc sua : ');
  readln(truong); if truong=0 then break;
  case truong of
    1: begin write('Ma nhan vien: '); readln(X.manv) end;
    2: begin write('Ho ten : '); readln(X.hoten) end;
    3: begin write('Luong chinh : '); readln(X.luong) end;
  end;
                end;
  seek(f1,k); write(f1,X);
                end
else writeln('Khong tim thay nhan vien co ma vua nhap');
close(f1); write('An Enter .....'); readln;
end;

procedure TimXoa;
var f1: tepnhanvien; a:dsnhanvien; i,n,k: integer;
    ma: string[6]; TimThay: boolean;
begin  assign(f1,'NVien.dat'); reset(f1);
n:=filesize(f1); for i:=1 to n do  read(f1,a[i]); close(f1);
write('Ban can tim nhan vien co ma de XOA: '); readln(ma);
TimThay:=false;
for i:=1 to n do if a[i].manv=ma then
                                begin TimThay:=True; k:=i; break; end;
if TimThay then  begin
    with a[k] do  begin
       writeln('Thong tin ve nhan vien vua tim thay: ');
       writeln('Ma nhan vien: ',manv);
       writeln('Ho va ten : ',hoten);
       writeln('Luong : ',luong:6:0);
                end;
    assign(f1,'NVien.dat'); rewrite(f1);
    for i:=1 to k-1 do write(f1,a[i]);
    for i:=k+1 to n do write(f1,a[i]); close(f1);
                end
else writeln('Khong tim thay nhan vien co ma vua nhap');
write('An Enter .....'); readln;
end;

procedure ChenBG;
var f1: tepnhanvien; x: nhanvien; a: dsnhanvien; i,n,j: integer;
begin assign(f1,'NVien.dat'); reset(f1);
n:=filesize(f1); for i:=1 to n do  read(f1,a[i]); close(f1);
```

```
for i:=1 to n do writeln(i,'. ',a[i].manv:6,' ',a[i].hoten:25);
write('Vao vi tri can CHEN ban ghi moi [1..',n,'] : ');
readln(j); if (j<1) or (j>n) then exit;
writeln('Hay nhap noi dung ban ghi moi:');
write('Ma nhan vien: '); readln(X.manv);
write('Ho ten : '); readln(X.hoten);
write('Luong chinh : '); readln(X.luong);
assign(f1,'NVien.dat'); rewrite(f1);
for i:=n downto j do a[i+1]:=a[i]; a[j]:=X;
for i:=1 to n+1 do write(f1,a[i]); close(f1);
write('Chen xong .....'); readln;
end;

procedure TimIn;
var f1: tepnhanvien; a: dsnhanvien;
    i,n,k,high,low,mid: integer; ma: string[6];
begin assign(f1,'NVienSX.dat'); reset(f1);
n:=filesize(f1); for i:=1 to n do  read(f1,a[i]); close(f1);
write('Ban can tim nhan vien co ma : '); readln(ma);
k:= 0; high:= n+1; low:= 0;
while (high-low > 1) and (k= 0) do
  begin mid:= (high+low) div 2;
        if a[mid].manv = ma then k:= mid
        else if ma > a[mid].manv then low:= mid
             else high:= mid;
  end;
if k>0 then  begin
  writeln('Thong tin ve nhan vien vua tim thay: ');
  writeln('Ma nhan vien: ',a[k].manv);
  writeln('Ho va ten : ',a[k].hoten);
  writeln('Luong : ',a[k].luong:6:0);
             end
else writeln('Khong tim thay nhan vien co ma vua nhap');
write('An Enter .....'); readln;
end;

BEGIN
repeat clrscr;
writeln('Chuong trinh Quan ly Nhan su:');
writeln('1. Tao tep CSDL moi NVien.dat va nhap');
writeln('2. Bo sung ban ghi vao cuoi tep NVien.dat');
writeln('3. Sap xep CSDL theo truong MANV tang dan');
writeln('4. Xem tep CSDL chua sap xep NVien.dat');
writeln('5. Xem tep CSDL da sap xep NVienSX.dat');
writeln('6. Tim kiem ban ghi theo MANV va sua');
writeln('7. Tim kiem ban ghi theo MANV va xoa');
writeln('8. Chen mot ban ghi moi');
writeln('9. Tim kiem ban ghi tren tep da sap xep va in');
writeln('10. Ket thuc chuong trinh'); writeln;
write('Hay chon mot chuc nang : '); readln(chon);
case chon of
1: Tao_BoSung(True);          2: Tao_BoSung(False);
3: SapXep;                    4: InCSDL('NVien.dat');
5: InCSDL('NVienSX.dat');     6: TimSua;
7: TimXoa;                    8: ChenBG;
9: TimIn;
```

```
end;
until chon=10;
end.
```

2. Tệp văn bản

Tệp văn bản là một tệp mà các phần tử của nó là các ký tự trong bảng mã ASCII. Các ký tự liên kết thành từng dòng, cuối mỗi dòng đánh dấu bởi hai ký tự có mã là 13 và 10. Turbo Pascal dùng tên chuẩn Text đặt tên cho kiểu tệp văn bản, do đó cách khai báo một tệp văn bản là `var f:text;`

Các thành phần trong các thao tác đọc ghi trên tệp văn bản có thể thuộc kiểu char, string, real, integer. Khi ghi các giá trị này lên tệp văn bản máy sẽ tự động chuyển sang dạng ký tự tương ứng. Do đó tệp văn bản chỉ có thể truy nhập tuần tự chứ không thể truy nhập một cách trực tiếp như tệp định kiểu. Tệp văn bản không phụ thuộc vào định nghĩa kiểu phần tử nên nó là công cụ giao tiếp chung giữa các chương trình, bất kỳ chương trình nào cũng có khả năng đọc dữ liệu từ tệp văn bản và xuất dữ liệu ra tệp văn bản. Các thiết bị như bàn phím, màn hình, máy in được xem như những dạng riêng của tệp văn bản. Việc soạn thảo một tệp văn bản có thể dùng một hệ soạn thảo bất kỳ, việc sửa chữa dữ liệu rất thuận tiện.

Để làm việc với tệp văn bản đầu tiên ta cũng phải đăng ký tên tệp với biến tệp f bằng thủ tục `assign(f,tentep)`, sau đó dùng thủ tục `reset(f)` để mở tệp đã có trên đĩa để đọc, hoặc dùng thủ tục `rewrite(f)` để mở một tệp mới để ghi.

Các hàm và thủ tục dùng với tệp văn bản:

• **Thủ tục Read(f, bien1, bien2, . . .) và Readln(f, bien1, bien2, . . .)** : đọc dữ liệu từ tệp f đưa vào các biến, các thủ tục này hoạt động giống như đọc dữ liệu từ bàn phím. Các biến kiểu char được đọc theo từng ký tự, các biến kiểu string được đọc theo đúng độ dài của nó hoặc gặp dấu xuống dòng, các biến số được đọc cho đến khi gặp dấu cách hay dấu xuống dòng. Lệnh readln đọc hết một dòng và con trỏ tệp xuống dòng kế tiếp, nếu còn dữ liệu thừa ở cuối dòng sẽ bị bỏ qua. Lệnh Read đọc dữ liệu từ tệp vào biến xong nhưng con trỏ tệp không xuống dòng.

• **Thủ tục Write(f, bt1, bt2, . . .) và Writeln(f, bt1, bt2, . . .)**: ghi giá trị các biểu thức (kiểu char, string, real, integer, boolean) lên tệp, thủ tục hoạt động giống như in lên màn hình, có thể tạo khuôn dạng in ra cho các biểu thức. Kết thúc ghi ra tệp cần đóng tệp bằng thủ tục Close(f). Lệnh Writeln(f): xuống dòng trên tệp văn bản.

• **Thủ tục Append(f)**: mở tệp văn bản để ghi bổ sung vào cuối tệp.

• **Hàm SeekEof(f)**: cho True nếu con trỏ tệp đang ở cuối tệp hoặc có thể nhảy qua các dấu cách, tab và dấu xuống dòng để đến cuối tệp; cho False nếu trái lại.

• **Hàm Eoln(f)**: cho True nếu con trỏ tệp đang ở cuối dòng, cho False nếu trái lại.

• **Hàm SeekEoln(f)**: cho True nếu con trỏ tệp đang ở cuối dòng hoặc có thể nhảy qua các dấu cách, tab để đến cuối dòng; cho False nếu trái lại.

• **Thủ tục SetTextBuf(var f: text; var Buf [; size: word])** tạo vùng đệm đọc ghi đĩa cho tệp văn bản ứng với biến tệp f. Kích thước ngầm định của vùng đệm cho biến tệp văn bản là 128 byte. Buf là một biến không định kiểu xác định vùng đệm mới, nó phải là biến toàn cục để không bị thu hồi khi đang thao tác vào ra tệp. Tuỳ chọn size

xác định kích thước vùng đệm, nếu không được dùng thì toàn bộ vùng nhớ của Buf sẽ được sử dụng. Cần gọi thủ tục này trước khi mở tệp. Khi chương trình cần đọc ghi nhiều lần thì việc tạo vùng đệm lớn sẽ làm tăng tốc độ truy nhập vì ít phải chuyển động đầu từ. Vùng đệm mới có hiệu lực cho đến khi gặp lệnh assign mới.

• **Thủ tục Flush(f)**: đẩy tất cả dữ liệu lưu trong vùng đệm ra đĩa từ.

Chương trình MangSo1.pas: đọc một mảng số thực từ tệp văn bản MangSo1.inp vào mảng a[1..n], sắp xếp mảng tăng dần, in mảng đã sắp xếp ra tệp MangSo1.out (mỗi dòng có 5 số).

```
uses crt;
var a: array[1..100] of real;
    f: text; i,j,n: integer; x: real;
begin clrscr; assign(f,'MangSo1.inp'); reset(f);
readln(f,n);
for i:=1 to n do read(f,a[i]); close(f);
for i:=1 to n-1 do for j:=i+1 to n do if a[i]>a[j] then
    begin x := a[i]; a[i]:= a[j]; a[j]:=x end;
assign(f,'MangSo1.out'); rewrite(f); j:=0;
writeln(f,n);
for i:=1 to n do begin write(f,a[i]:10:5,'  '); j:=j+1;
                       if j mod 5=0 then writeln(f); end;
close(f); write('An Enter ....'); readln;
end.
```

Cấu trúc tệp MangSo1.inp: dòng đầu là n (số phần tử của mảng), các dòng sau là các số thực viết cách nhau bởi dấu cách, tab hay dấu xuống dòng (có tất cả đúng n số). Ví dụ tệp MangSo1.inp:

```
30
8.1   4.2   3.4   2.3   1.5   5.0   6    7     23    56
22.9  81.1  9.6   12.3  7.3   6.7   9.5  11.6  54.3  35.76
123   231   2.2   87    453   765   23   111   993   192
```

Chương trình MangSo2.pas làm nhiệm vụ như chương trình MangSo1.pas nhưng không biết trước số lượng số thực có trong tệp (tức là không biết trước n). Tệp MangSo2.inp có dạng:

```
8.1   4.2   3.4   2.3   1.5   5.0   6    7     23    56
22.9  81.1  9.6   12.3  7.3   6.7   9.5  11.6  54.3  35.76
123   231   2.2   87    453   765   23   111   993   192
```

Trong trường hợp cần đọc các giá trị số cho đến hết dòng hoặc hết tệp mà không biết trước số lượng các giá trị cần đọc ta dùng các hàm SeekEof(f) hoặc SeekEoln(f) để tránh đọc sót hoặc đọc vào nơi không có dữ liệu.

```
uses crt;
var a: array[1..100] of real;
    f: text; i,j,n: integer; x: real;
begin clrscr;
assign(f,'MangSo2.inp'); reset(f); n:=0;
while not SeekEof(f) do begin inc(n); read(f,a[n]) end; close(f);
for i:=1 to n-1 do for j:=i+1 to n do if a[i]>a[j] then
    begin x := a[i]; a[i]:= a[j]; a[j]:=x end;
assign(f,'MangSo2.out'); rewrite(f); j:=0;
writeln(f,n);
```

```
for i:=1 to n do begin write(f,a[i]:10:5,'  '); j:=j+1;
                      if j mod 5=0 then writeln(f); end;
close(f); write('An Enter ....'); readln;
end.
```

Chương trình MaHoa.pas dùng để mã hoá một tệp văn bản và sau đó giải mã. Chức năng 1 của chương trình: vào tên tệp văn bản cần mã hoá (ví dụ Vb1.txt), vào tên tệp chứa kết quả mã hoá (ví dụ Vb2.txt), chương trình đọc từng dòng văn bản từ tệp, mã hoá từng ký tự của dòng vừa đọc theo bảng chuyển mã, ghi dòng đã mã vào tệp Vb2.txt. Chức năng 2 của chương trình: vào tên tệp cần giải mã (ví dụ Vb2.txt), vào tên tệp chứa kết quả giải mã (ví dụ Vb3.txt), chương trình tự động giải mã. Nếu mã hoá và giải mã đúng thì tệp Vb1.txt phải trùng với tệp Vb3.txt.

```
Uses crt;
Var goc,ma,s: string; i,j,n,k,m,chon: integer;
    f1,f2: text; tf1,tf2: string[30];
Begin clrscr;
goc :='abcdefghijklmnopqrstuvwxyz1234567890AEIOU ';
ma  :='3oUn4Opced9E10kmlw2Ijqrabxyfsz5gtA6u7hv8iπ';
k:= length(goc);
writeln('1. Ma hoa 2. Giai ma. Chon: '); readln(chon);
write('Vao tep1: '); readln(tf1); assign(f1,tf1); reset(f1);
write('Tep 2: '); readln(tf2); assign(f2,tf2); rewrite(f2);
while not eof(f1) do
   begin readln(f1,s); m:= length(s); writeln(s);
         for i:=1 to m do
            case chon of
             1: for j:=1 to k do if goc[j]=s[i] then
                                 begin s[i]:=ma[j]; break end;
             2: for j:=1 to k do if ma[j]=s[i] then
                                 begin s[i]:=goc[j]; break end;
            end;
         writeln(f2,s); writeln(s); readln;
   end;
close(f1); close(f2); write('Xong'); readln; end.
```

Chương trình VBanGB.pas: đọc dữ liệu các bản ghi soạn dưới dạng tệp văn bản Bank.dat và đưa vào mảng a[1..n], sắp xếp mảng theo mã tài khoản tăng dần nhờ thuật toán nổi bọt và in mảng a ra màn hình. Bản ghi có 4 trường: mã tài khoản, họ tên chủ tài khoản, số dư tiền gửi, địa chỉ chủ tài khoản. Mẫu tệp Bank.dat:

```
447,Bui The Tam,560000000.0,5-Doi Can-Ba dinh,
987,Nguyen Van Thanh,150000000.0,35-Tran Hung Dao-Hoan kiem,
112,Tran Quang Minh,4566,179-Trang Tien-Hoan kiem,
367,Vu Dang Binh,2222222.222,35-Nha tho-Bac ninh,
```

Cấu trúc của tệp: mỗi dòng ứng với một bản ghi, dữ liệu các trường tách nhau bởi dấu phẩy, ký tự đầu tiên của trường sau viết liền ngay dấu phẩy, sau trường cuối cùng của một dòng cũng có dấu phẩy. Có thể dùng hệ soạn thảo văn bản bất kỳ để tạo, chỉnh sửa dữ liệu, thêm bớt bản ghi của tệp Bank.dat.

Thủ tục XuLyDong(d,x): tách dòng văn bản d thành các trường của bản ghi x, trường thứ i lưu vào xâu s[i], nếu xâu s[i] ứng với trường số thì phải chuyển xâu sang

số nhờ thủ tục Val . Đối với các bài toán khác ta cần sửa thủ tục XuLyDong() cho phù hợp (thêm bớt tên trường và chuyển đổi kiểu từ xâu sang số).

```
uses crt;
const nmax = 100;
type taikhoan = record
                   matk: integer;
                   hoten: string[25];
                   tien: real;
                   diachi: string[30];
                end;
     dstaikhoan= array[1..nmax] of taikhoan;
     mangxau= array[1..10] of string[30];
var  a:dstaikhoan; x:taikhoan; i,j,n:Integer;  f:text; d:string;

procedure XuLyDong(d:string; var x:taikhoan);
var t:string[30]; j,dem,k:integer; s:mangxau;
begin for i:=1 to 10 do s[i]:=''; k:=1; t := ''; dem:=0;
for j:=1 to length(d) do
 begin if d[j]<> ','then begin t:=t+d[j]; inc(dem) end
                     else begin t[0]:= chr(dem); s[k]:=t;
                                k:=k+1; t:=''; dem:=0 end;
 end;
val(s[1],x.matk,j); x.hoten:=s[2];
val(s[3],x.tien,j); x.diachi:=s[4];
end;

begin clrscr;
assign(f,'bank.dat'); reset(f); n:=0;
while not eof(f) do
   begin readln(f,d); XuLyDong(d,x); inc(n); a[n]:=x  end;
close(f);
for i:=1 to n do with a[i] do
   writeln(matk:6,' ',hoten:25,'   ',tien:10:0,'   ',diachi);

for i:=1 to n-1 do for j:=1 to n-i do
  if a[j].matk > a[j+1].matk then
     begin  x:= a[j]; a[j]:= a[j+1]; a[j+1]:= x  end;
writeln('Mang da sap xep : ');
for i:=1 to n do with a[i] do
   writeln(matk:6,' ',hoten:25,'   ',tien:10:0,'   ',diachi);
readln;
end.
```

Chương trình VungDem.pas : tạo vùng đệm 4 KB để đọc / ghi tệp văn bản.

```
uses crt;
var f: text;  ch: char;
    buf: array[1..4096] of char;
begin  clrscr; assign(f,'VBan.txt');
settextbuf(f, buf);  reset(f);
while not eof(f) do
begin
read(f,ch);  write(ch); ;
end; readln
end.
```

3. Tệp không định kiểu

Tệp không định kiểu dùng để truy nhập các tệp mà không cần quan tâm đến kiểu dữ liệu (chẳng hạn khi sao chép tệp hay truyền dữ liệu) nhằm làm tăng tốc độ truy nhập. Cách khai báo tệp không định kiểu: `var f:file;`

Thực chất tệp không định kiểu giống như tệp định kiểu, trong đó mỗi phần tử là một đoạn các byte do người dùng tự quy định, do đó nó dùng tương đối giống tệp định kiểu. Trước khi dùng tệp ta cũng phải gán tên tệp cho biến tệp bằng lệnh assign(f, tentep).

Các hàm và thủ tục dùng với tệp không định kiểu:

• **Thủ tục Reset(var f: file; recsize: word)**: mở tệp và con trỏ tệp đặt ở đầu tệp. f là biến tệp đã được liên kết với một tệp bằng lệnh assign. RecSize là biểu thức tuỳ chọn kiểu word qui định kích thước theo byte của phần tử (record) được dùng trong truyền dữ liệu, nếu vắng RecSize thì kích thước ngầm định của một phần tử là 128 byte. Con trỏ tệp di chuyển theo kích thước của một phần tử, do đó nếu độ dài tệp không chia hết cho kích thước một phần tử thì một số byte ở cuối tệp sẽ bị bỏ qua. Để tránh điều này ta thường cho kích thước phần tử là 1 byte (recsize=1).

• **Thủ tục Rewrite(var f: file; recsize: word)**: tạo một tệp mới để ghi, ý nghĩa của recsize như trong thủ tục reset.

• **Thủ tục BlockRead(var f: file; var buf; count : word; var result: word)**: f là biến tệp không định kiểu, Buf là biến kiểu bất kỳ đóng vai trò vùng đệm chứa dữ liệu đọc từ tệp. BlockRead đọc Count phần tử (hoặc ít hơn) từ tệp f vào bộ nhớ bắt đầu từ byte đầu tiên của Buf, việc đọc tệp định kiểu tiến hành theo từng khối dữ liệu. Số lượng các phần tử hoàn chỉnh được đọc (<=Count) trả về cho tham số tuỳ chọn Result. Nếu không dùng Result, lỗi vào ra xảy ra khi số lượng phần tử được đọc không bằng Count. Toàn bộ khối được truyền chiếm Count * RecSize byte, trong đó RecSize là kích thước phần tử được chỉ định khi mở tệp. Nếu Count * RecSize lớn hơn 65535 (64 KB) sẽ sinh ra lỗi.

Result là tham số tuỳ chọn. Nếu toàn bộ khối được truyền thì Result sẽ bằng với Count. Trái lại, nếu Result < Count thì ta đã gặp cuối tệp trước khi việc truyền dữ liệu được hoàn tất. Trong trường hợp nếu kích thước phần tử lớn hơn 1, Result sẽ trả về số lượng phần tử hoàn chỉnh được đọc. Dấu hiệu Result = 0 thường dùng để kiểm tra việc đọc đã hoàn tất chưa. Sau lệnh BlockRead con trỏ tệp tiến lên Result phần tử. Khi dùng {SI-} IOResult trả về 0 nếu thủ tục được thực hiện thành công, trái lại nó cho một mã khác 0.

• **Thủ tục BlockWrite(var f: file; var buf; count: word; var result: word)**: ghi Count (hoặc ít hơn) phần tử từ bộ nhớ vào tệp f bắt đầu từ byte đầu tiên của Buf (buf là biến có kiểu bất kỳ đóng vai trò vùng đệm chứa dữ liệu sẽ ghi lên tệp). Result: số lượng phần tử thực sự được ghi lên tệp. Nếu không dùng Result, lỗi vào ra xuất hiện khi số lượng phần tử ghi lên tệp không bằng Count. Toàn bộ khối được ghi gồm Count * RecSize byte. Nếu toàn bộ khối được truyền thì Result = Count. Trái lại Result < Count, chứng tỏ đĩa đã đầy trước khi việc truyền được hoàn tất. Sau lệnh này con trỏ tệp cũng tiến lên Result phần tử.

Chương trình **CopyFile.pas**: sao chép một tệp bất kỳ sang một tệp khác, các tệp được khai báo dưới dạng không định kiểu, kích thước một phần tử RecSize = 1, kích thước vùng đệm là 2048 byte (2 KB). Thuật toán: đọc từng khối của tệp nguồn rồi chép khối sang tệp đích.

```
program CopyFile;
var FromF, ToF: file;   NumRead, NumWritten: Word;
    Buf: array[1..2048] of Char;
begin  Assign(FromF, ParamStr(1)); { mo tep nguon }
Reset(FromF, 1);  { kich thuoc phan tu = 1 }
Assign(ToF, ParamStr(2)); { mo tep dich }
Rewrite(ToF, 1);  { kich thuoc phan tu = 1 }
Writeln('Copying ', FileSize(FromF), ' bytes...');
repeat
     BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
     BlockWrite(ToF, Buf, NumRead, NumWritten);
until (NumRead = 0) or (NumWritten <> NumRead);
Close(FromF); Close(ToF);
end.
```

Chương trình cần dịch ra tệp CopyFile.exe, sau đó gõ tên chương trình tại dấu nhắc lệnh của DOS. Ví dụ để sao chép tệp Turbo.exe thành tệp LapTrinh.exe ta dùng lệnh: `c:\>copyfile turbo.exe laptrinh.exe` ↵ . Hai tên tệp sau tên chương trình gọi là các tham số dòng lệnh. Chương trình sử dụng hàm ParamStr().

• **Hàm ParamStr(i: word): string** trả về tham số dòng lệnh thứ i hoặc trả về xâu rỗng nếu i > ParamCount (số lượng tham số truyền cho chương trình trên dòng lệnh, kiểu word). ParamStr(0) trả về đường dẫn và tên tệp của chương trình đang thực hiện.

4. Kiểm tra lỗi vào ra

Turbo Pascal dùng các định hướng biên dịch sau để tắt / mở việc kiểm tra lỗi trong quá trình vào/ra tệp:

• **{$I+}**: mở việc kiểm tra lỗi đối với các thủ tục vào/ra (chế độ ngầm định). Khi gặp lỗi vào/ra chương trình sẽ thông báo lỗi và dừng lại

• **{$I-}**: không kiểm tra lỗi vào/ra. Khi gặp lỗi chương trình không dừng lại, việc xử lý lỗi dành cho người dùng bằng cách sử dụng hàm **IOResult: integer**. Hàm IOResult trả về một số nguyên cho biết trạng thái của thao tác vào/ra: 0- không có lỗi, 2- không tìm thấy tệp, 3- không tìm thấy đường dẫn, 4- mở quá nhiều tệp, 5- tệp không thể truy nhập . . .

Chương trình **FileSize.pas** cho kích thước tệp, tên tệp nhập vào theo tham số dòng lệnh. Chương trình cần dịch ra tệp FileSize.exe và chạy dưới dấu nhắc của DOS, ví dụ xem kích thước tệp Turbo.exe dùng lệnh: `c:\>filesize turbo.exe` ↵

```
var F: file of Byte;
begin Assign(F, ParamStr(1));  {$I-}  Reset(F);  {$I+}
if IOResult = 0 then
          Writeln('Kich thuoc tep theo byte: ', FileSize(F))
else  Writeln('Khong tim thay tep');
end.
```

Đoạn chương trình sau cho phép mở một tệp định kiểu để nhập các bản ghi (nếu chưa có thì tạo tệp mới, nếu tệp đã có thì ghi bổ sung):

```
{$I-}  Reset(f);  {$I+}
if IOResult <> 0 then rewrite(f) else seek(f, filesize(f)); ...
```

Bài tập

1. Xây dựng kiểu dữ liệu cấu trúc hộ_điện gồm trường: mã khách hàng, tên chủ hộ dùng điện, địa chỉ, chỉ số công tơ đầu tháng, chỉ số công tơ cuối tháng, số Kwh dùng trong tháng, thành tiền. Tệp CSDL là Dien.dat. Menu chính của chương trình gồm các chức năng:

- Nhập một khách mới vào CSDL với các thông tin: mã khách hàng (không trùng mã với những người đã có trong CSDL), họ tên chủ hộ, địa chỉ, chỉ số công tơ đầu tháng, chỉ số công tơ cuối tháng. Trường Số Kwh sử dụng = chỉ số công tơ cuối tháng - chỉ số công tơ đầu tháng. Trường Thành tiền tính theo quy tắc: nếu sử dụng từ 1 đến 100 số thì đơn giá là 1000 đồng / số, từ 101 đến 150 số đơn giá là 1300 đồng / số, từ 151 đến 200 đơn giá là 1600 đồng / số, trên 200 số đơn giá là 2000 đồng / số.
- Xem toàn bộ các khách hàng.
- Tìm kiếm và sửa: nhập vào một mã khách hàng, tìm trong CSDL bản ghi có mã vừa nhập, nếu tìm thấy thì cho hiện bản ghi này lên màn hình và cho phép sửa các trường của bản ghi tìm được, sửa xong lại ghi vào tệp CSDL.
- Tìm kiếm và xoá: nhập vào một mã khách hàng, tìm trong CSDL bản ghi có mã vừa nhập, nếu tìm thấy thì cho hiện bản ghi này lên màn hình và cho phép xoá bản ghi này khỏi CSDL.
- Tìm kiếm và chèn: nhập vào một mã khách hàng, tìm trong CSDL bản ghi có mã vừa nhập, nếu tìm thấy thì cho hiện bản ghi này lên màn hình và cho phép chèn vào sau bản ghi này một bản ghi mới.
- Tìm kiếm và in: nhập vào một mã khách hàng, tìm trong CSDL bản ghi có mã vừa nhập, nếu tìm thấy thì cho hiện bản ghi này lên màn hình và cho phép in bản ghi này ra giấy.
- In bảng tiền điện đã sắp xếp theo Mã khách hàng với các cột: mã khách hàng, họ tên, chỉ số đầu, chỉ số cuối, số Kwh sử dụng, thành tiền. Cuối bảng có ghi Tổng số tiền thu của toàn chi nhánh điện.

2. Lập chương trình bán vé máy bay cho một hãng hàng không với các kiểu dữ liệu khai báo như sau:

```
const nmax = 30;
type chuyenbay = record
                   dich, ngay, gio: string[15] ;
                   cho: array[1..100] of byte;
                 end;
mangchuyenbay = array[1..nmax] of chuyenbay ;
tepchuyenbay = file of chuyen bay;
```

Thông tin về một chuyến bay gồm có: đích của chuyến bay (ví dụ Saigon), ngày bay (25/12/2004), giờ bay (15.30), mảng tình trạng đã bán vé các chỗ ngồi trên máy bay (cho[j] = 1 nếu ghế j đã bán vé, cho[j] = 0 nếu ghế j còn trống).

Chương trình được điều khiển theo menu với các chức năng sau:

1) *Nhập thông tin về một chuyến bay mới* vào tệp Ve.dat. Mỗi chuyến bay cần nhập: đích của chuyến bay, ngày bay, giờ bay. Tất cả các phần tử của mảng **cho** đều

gán bằng 0 (các ghế đều trống).

2) *In ra màn hình tình trạng hiện tại của cơ sở dữ liệu* Ve.dat: các trường in theo cột, mảng **cho** chỉ in 30 phần tử đầu gồm các số 0 và 1 viết liền nhau.

3) *Đặt chỗ và in vé cho khách hàng.* Khi một khách hàng đến mua vé, chương trình cho hiện danh sách các chuyến bay (như chức năng 2), sau đó chương trình yêu cầu nhập vào Tên khách hàng, chuyến bay (ví dụ HUE), ngày bay và giờ bay khách yêu cầu, số ghế ngồi mà khách muốn (ví dụ, số ghế 25). Tiếp theo máy kiểm tra số ghế 25 của chuyến bay này đã có ai đặt chưa. Nếu chỗ 25 đã có người đặt (cho[25]=1) thì hiện mảng **cho** lên màn hình để xem còn chỗ trống hay không và ta vào lại một chỗ khác. Còn nếu cho[25]=0 thì ta in vé cho khách hàng theo mẫu:

Họ và tên khách hàng : *Nguyễn Văn Thanh*
Chuyến bay từ Hà nội đi *Huế*
Ngày bay : *27/12/2004* Giờ bay : *12 giờ 30 phút*
Số ghế ngồi : 25 Giá tiền : *1.000.000* đồng

và gán Cho[25]=1 (đã có người đặt). Biết rằng giá vé đi Huế 1 triệu đồng, đi Đà nẵng 1,2 triệu, đi Sài gòn 1,8 triệu đồng.

4) *Xoá tất cả các chuyến bay đã bay.* Nhập vào ngày và giờ hiện tại. Xoá tất cả các chuyến bay có thời gian nhỏ hơn ngày giờ hiện tại ra khỏi cơ sở dữ liệu Ve.dat.

3. Để giúp cho giáo viên phổ thông tính điểm trung bình cho một môn học của một lớp cần khai dữ liệu kiểu cấu trúc **hsinh** như sau:

```
type hsinh = record
                stt: integer; hten: string[25];
                heso1,heso2: string[6]; tbktra,thi,tbmon: real;
             end;
```

Thông tin về mỗi học sinh gồm có: số thứ tự trong sổ điểm chính, họ tên học sinh, các điểm hệ số 1 (là một xâu ký tự biểu thị nhiều nhất 6 điểm, điểm mười ký hiệu là chữ 'M', ví dụ '80M5' là 4 điểm 8, 0, 10, 5), các điểm hệ số 2 (nhiều nhất là 6 điểm và cũng ký hiệu tương tự), điểm trung bình kiểm tra, điểm thi cuối học kỳ, điểm trung bình môn. Điểm trung bình kiểm tra (tbktra) được tính bằng cách cộng các điểm hệ số 1 với các điểm hệ số 2 có nhân đôi và sau đó chia trung bình. Ví dụ : các điểm hệ số 1 là 8, 0, 10, 5 và các điểm hệ số 2 là 1, 10, 9, 4 thì điểm trung bình kiểm tra là (8 + 0 + 10 + 5 + 2*1 + 2*10 + 2*9 + 2*4) / 12. Điểm trung bình môn (tbmon) = (2*tbktra + điểm thi cuối học kỳ) / 3.

Thông tin về các học sinh tổ chức dưới dạng tệp văn bản HSinh.dat. Mỗi dòng của tệp văn bản ứng với một học sinh gồm có các thông tin: số thứ tự họ tên, điểm hệ số 1, điểm hệ số 2, điểm thi (thông tin mỗi trường này cách nhau bởi dấu phẩy, sau trường cuối cũng có dấu phẩy, kí tự đầu tiên của một trường viết liền ngay dấu phẩy). Mẫu tệp HSinh.dat:

```
1,Bui The Tam,5M1M,M34M7,8.5,
2,Nguyen Van Thanh,765M,M8765,9.5,
3,Tran Minh,MM5,M087,5,
4,Vu Ngoc Phan,M87,6MM34,7.75,
```

Lập trình nhập các thông tin ban đầu của từng học sinh, tính hai trường Tbktra và Tbmon. In bảng kết quả học tập của lớp dưới dạng tệp văn bản gồm 7 cột ứng với 7 trường.

Chương 7

Unit Crt

Unit là một tập hợp các hằng số, các kiểu dữ liệu, các biến, các thủ tục và hàm. Mỗi Unit xem như một thư viện chương trình. Turbo Pascal có sẵn các unit chuẩn: unit System, unit Crt, unit Dos, unit Printer, unit Overlay, unit Graph, unit Turbo3, unit Graph3, unit Windos. Năm unit đầu tiên chứa trong tệp Turbo.tpl, tệp này được tải vào bộ nhớ ngay khi Turbo Pascal khởi động. Ngoài các unit chuẩn có sẵn người dùng có thể tự mình lập các unit riêng.

Trong chương này sẽ nghiên cứu unit Crt, unit này dùng để điều khiển việc in thông tin ra màn hình, điều khiển bàn phím và sử dụng âm thanh. Khi dùng các hàm và thủ tục của unit Crt ở đầu chương trình cần có khai báo: `uses crt;`

1. Điều khiển in thông tin ra màn hình ở chế độ văn bản

Trong chế độ văn bản màn hình chuẩn hiện nay có 80 cột và 25 dòng, góc trên bên trái màn hình có toạ độ (1,1), trục x hướng sang phải và có giá trị từ 1 đến 80, trục y hướng xuống dưới và có giá trị từ 1 đến 25. Trên màn hình có một điểm sáng nhấp nháy gọi là con trỏ màn hình. Việc in thông tin bởi lệnh write bắt đầu từ vị trí con trỏ, in xong một ký tự con trỏ tự động chuyển tới vị trí kế tiếp. Thông tin được nhập vào từ bàn phím bởi lệnh read cũng hiện lên màn hình bắt đầu từ vị trí con trỏ.

Cửa sổ văn bản và các hàm, thủ tục liên quan

• **Thủ tục Window(x1, y1, x2, y2: byte)**: thiết lập một cửa sổ mới để trình bày. (x1,y1) là toạ độ góc trên bên trái và (x2,y2) là toạ độ góc dưới bên phải cửa sổ. Khi một cửa sổ mới được thiết lập thì mọi kết xuất ra màn hình đều nằm trong cửa sổ mới, mọi giá trị toạ độ đều là tương đối đối với cửa sổ hiện hành (góc trên trái cửa sổ có toạ độ (1,1)). Cửa sổ hiện hành có hiệu lực cho đến khi định nghĩa một cửa sổ khác. Lời gọi window(1,1,80,25) sẽ quay về chế độ toàn màn hình (cửa sổ ngầm định).

• **Biến WindMin và WindMax** lưu trữ toạ độ màn hình của cửa sổ hiện hành, các biến này được đặt bởi lời gọi thủ tục Window. WinMin xác định góc trên trái, WinMax xác định góc dưới bên phải. Toạ độ x lưu ở byte thấp, toạ độ y lưu ở byte cao. Ví dụ Lo(WinMin) cho toạ độ x của cạnh trái, Hi(WinMax) cho toạ độ y của cạnh dưới. Góc trên trái của màn hình có toạ độ (x,y) = (0,0). Chú ý, với các toạ độ truyền cho các thủ tục Window và Gotoxy, góc trên trái là (1,1).

Hàm Lo(x): byte trả về byte thấp của x, còn hàm *Hi(x): byte* trả về byte cao của x, trong cả hai hàm x là biểu thức kiểu integer hay word.

• **Thủ tục Crlscr** : xoá tất cả các ký tự trên cửa sổ hiện hành, đưa con trỏ về toạ độ (1,1), đặt màu nền của cửa sổ theo lệnh TextBackGround trước đó.

• **Thủ tục ClrEol** : xoá các ký tự từ vị trí hiện tại của con trỏ cho tới cuối dòng trong cửa sổ hiện hành.

• **Thủ tục DelLine**: xoá dòng chứa con trỏ trong cửa sổ hiện hành, các dòng ở dưới dịch lên trên một dòng, một dòng trống với màu nền hiện hành thêm vào đáy cửa sổ.

• **Thủ tục InsLine**: chèn một dòng trống với màu nền hiện hành vào vị trí con trỏ hiện tại, các dòng ở dưới sẽ đẩy xuống một dong.

• **Thủ tục Gotoxy(x,y: byte)**: đưa con trỏ về cột x dòng y của cửa sổ hiện tại.

• **Hàm WhereX: byte** cho toạ độ x của con trỏ trong cửa sổ hiện tại.

• **Hàm WhereY: byte** cho toạ độ y của con trỏ trong cửa sổ hiện tại.

Màu chữ và màu nền. Mỗi ký tự trên màn hình tương ứng với hai giá trị màu: màu chữ và màu nền (màu của hình chữ nhật phía sau ký tự). Unit Crt định nghĩa các hằng với các tên màu tương ứng:

giá trị	tên hằng	màu	hệ 2	giá trị	tên hằng	màu	hệ 2
0	Black	đen	000	8	DarkGray	xám tối	1000
1	Blue	xanh	001	9	LightBlue	xanh sáng	1001
2	Green	xanh lá cây	010	10	LightGreen	xanh lá cây sáng	1010
3	Cyan	xanh lơ	011	11	LightCyan	xanh lơ sáng	1011
4	Red	đỏ	100	12	LightRed	đỏ sáng	1100
5	Magenta	tím	101	13	LightMagenta	tím sáng	1101
6	Brown	nâu	110	14	Yellow	vàng	1110
7	LightGray	xám sáng	111	15	White	trắng	1111

• **Thủ tục TextBackGround(color: byte)**: đặt màu nền, color có giá trị từ 0 đến 7. Muốn màu nền loang đều trên màn hình thì sau lệnh này ta phải có lệnh Clrscr.

• **Thủ tục TextColor(color: byte)**: đặt màu chữ, color có giá trị từ 0 đến 15. Các giá trị màu được đặt có hiệu lực với các lệnh in sau đó cho đến khi gặp một lệnh đặt màu mới. Màu nền ngầm định là 0, màu chữ ngầm định là 7.

Thuộc tính màu lưu trong một byte có cấu trúc như sau: bit 7 là thuộc tính nhấp nháy, ba bit tiếp theo (6, 5, 4) là màu nền chứa được 8 giá trị, bốn bit đầu tiên (3, 2, 1, 0) là màu chữ chứa được 16 giá trị. Vì 1000 000 (hệ 2) = 128 hệ 10, nên hằng Blink trong unit Crt định nghĩa là 128. Unit Crt dùng biến **TextAttr** : byte để cất giữ thuộc tính màu theo cấu trúc như trên. Chẳng hạn nếu TextAttr hiện có giá trị bằng 61 thì màu nền là Cyan, màu chữ là LightMagenta, vì 61 = 3*16+13. Ngược lại ta cũng có thể gán thuộc tính màu cho biến này để đặt lại màu chữ và màu nền, lệnh `TextAttr :=1*16+15+128` sẽ đặt màu nền là Blue, màu chữ là White và nhấp nháy.

• **Thủ tục NormVideo**: khôi phục biến TextAttr theo giá trị mà nó có khi chương trình bắt đầu chạy.

• **Thủ tục HighVideo**: cho bit 3 của TextAttr bằng 1, tức là chuyển các màu 0-7 thành các màu 8-15 tương ứng.

• **Thủ tục LowVideo**: chon bit 3 của TextAttr bằng 0, tức là chuyển các màu 8-15 thành các màu 0-7 tương ứng.

Chương trình **TLuong.pas**: nhập dữ liệu về một nhân viên trong một cửa sổ (mã,

họ tên, hệ số lương, hệ số phụ cấp trách nhiệm), in kết quả tính lương ở cửa sổ thứ hai (lương chính = 290000 * hệ số lương, phụ cấp = 290000*hệ số phụ cấp, bảo hiểm xã hội = 5% lương chính, bảo hiểm y tế = 1% lương chính, số tiền được lĩnh), về lại cửa sổ thứ nhất để nhập người tiếp theo. Chương trình minh hoạ cách mở nhiều cửa sổ và chuyển đổi cửa sổ làm việc. Cuối chương trình cần khôi phục cửa sổ ngầm định của DOS và thuộc tính văn bản ngầm định. Thủ tục Khung(x1,y1,x2,y2): vẽ một khung nét kép trên màn hình với toạ độ góc trên trái là (x1,y1), toạ độ góc dưới bên phải là (x2,y2). Các cửa sổ trên màn hình có đóng khung bên ngoài.

```
uses crt;
type nhanvien = record
                  manv: string[6];
                  hoten: string[25];
                  hesoluong, hesopc: real;
                  luong,phucap,bhxh,bhyt,tien: real;
                end;

procedure Khung(x1,y1,x2,y2: byte);
var i: integer;
begin
for i:=x1+1 to x2-1 do
   begin gotoxy(i,y1); write(chr(205));
         gotoxy(i,y2); write(chr(205)) end;
for i:=y1+1 to y2-1 do
   begin gotoxy(x1,i); write(chr(186));
         gotoxy(x2,i); write(chr(186)) end;
gotoxy(x1,y1); write(chr(201)); gotoxy(x2,y1); write(chr(187));
gotoxy(x1,y2); write(chr(200)); gotoxy(x2,y2); write(chr(188));
end;
var x: nhanvien; min: real; tt:char;

begin textbackground(1); textcolor(15); clrscr;
min:=290000; Khung(1,1,39,9); Khung(41,11,80,24); tt:='C';
while upcase(tt)='C' do begin

window(2,2,38,8); clrscr;
gotoxy(9,1); write('DU LIEU NHAP VAO');
gotoxy(3,2); write('Vao ma nhan vien : '); readln(x.manv);
gotoxy(3,3); write('Vao ho ten : '); readln(x.hoten);
gotoxy(3,4); write('He so luong : '); readln(x.hesoluong);
gotoxy(3,5); write('He so phu cap trach nhiem : '); readln(x.hesopc);
gotoxy(3,6); write('An Enter xem ket qua tinh...'); readkey;

window(42,12,79,23);
gotoxy(9,1); write('KET QUA TINH LUONG');
gotoxy(2,2); write('Ma nhan vien : ',x.manv);
gotoxy(2,3); write('Ho ten : ',x.hoten);
gotoxy(2,4); write('He so luong : ',x.hesoluong:1:2);
gotoxy(2,5); write('He so phu cap trach nhiem : ',x.hesopc:1:2);
x.luong := min*x.hesoluong;
x.phucap := min*x.hesopc;
x.bhxh := 0.05* x.luong;
x.bhyt := 0.01* x.luong;
x.tien := x.luong+x.phucap-x.bhxh-x.bhyt;
gotoxy(2,6); write('Luong chinh : ',x.luong:7:0);
gotoxy(2,7); write('Phu cap trach nhiem : ',x.phucap:7:0);
```

```
gotoxy(2,8); write('Bao hiem xa hoi : ',x.bhxh:7:0);
gotoxy(2,9); write('Bao hiem y te : ',x.bhyt:7:0);
gotoxy(2,10); write('Tien thuc linh : ',x.tien:7:0);
gotoxy(2,11); write('Co nhap tiep khong C/K ?'); tt:= readkey;
if upcase(tt)<>'C' then break; clrscr;
                  end;
window(1,1,80,25); TextAttr:= 7; clrscr;
end.
```

Chương trình **Mau.pas** minh hoạ cách dùng các hàm và thủ tục về màu sắc.

```
uses crt;
begin  clrscr;
writeln('TextAttr sau khi khoi dong chuong trinh = ',TextAttr);{7}
TextAttr := $B6; {= 1011 0110 he 2} writeln;
writeln('Mau chu Brow nhap nhay, mau nen Cyan');
HighVideo;  { TextAttr = 1011 1110 he 2 } writeln;
writeln('Mau chu Yellow nhap nhay, mau nen Cyan');
TextAttr := $2D; {= 0010 1101 he 2} writeln;
writeln('Mau chu LightMagenta khong nhay, mau nen Green');
LowVideo; { TextAttr = 0010 0101 he 2} writeln;
writeln('Mau chu Magenta khong nhay, mau nen Green');
NormVideo; { TextAttr = 0000 0111 he 2 = 7 he 10} writeln;
writeln('Mau chu LightGray, nen Black, TextAttr= ',TextAttr);
readln;
end.
```

Các chế độ màn hình

• **Thủ tục TextMode(mode: word)**: đặt lại chế độ màn hình, toàn bộ màn hình cũ bị xoá, về lại chế độ toàn màn hình, con trỏ về góc trên bên trái, biến TextAttr được đặt lại theo giá trị nguyên thuỷ. Các giá trị của Mode:

Tên hằng	Giá trị	Mô tả
BW40	0	40 cột 25 dòng, đen/trắng trên màn hình màu
C40	1	40 cột 25 dòng, chế độ màu trên màn hình màu
BW80	2	80 cột 25 dòng, đen/trắng trên màn hình màu
C80	3	80 cột 25 dòng, chế độ màu trên màn hình màu
Mono	7	80 cột 25 dòng, đen/trắng trên màn hình đơn sắc
Font8x8	256	đặt chế độ 50 dòng cho màn hình VGA

Giá trị ngầm định cho các màn hình hiện nay là C80. Giá trị Font8x8 dùng kết hợp với các giá trị khác, ví dụ TextMode(C40+Font8x8) sẽ đặt chế độ màn hình 40 cột, 50 dòng, 16 màu trên màn hình VGA. Biến LastMode lưu lại giá trị mode của lần gọi TextMode gần nhất.

Chương trình **Mode.pas** trình diễn tất cả các chế độ màn hình. Thủ tục Print() lấp đầy màn hình bằng một ký tự.

```
uses crt;
procedure Print(m,n: byte; ch: char);
var i,j: integer;
begin clrscr;
```

```
for i:=1 to m-1 do for j:=1 to n do begin gotoxy(j,i); write(ch);
end;
for j:=1 to n-1 do begin gotoxy(j,m); write(ch) end;
gotoxy(1,1);
end;
var OldMode: byte;

begin clrscr;
writeln('Lastmode = ',LastMode,' TextAttr = ',TextAttr); readkey;
OldMode:=Lastmode;  TextAttr:= 3*16+12; Print(25,80,'A'); readkey;
TextMode(C80+Font8x8); TextAttr:=1*16+ 15; Print(50,80,'B'); readkey;
TextMode(Mono); Print(25,80,'C'); readkey;
TextMode(C40+Font8x8);TextAttr:=1*16+13; Print(50,40,'D'); readkey;
TextMode(BW80); Print(25,80,'E'); readkey;
TextMode(BW80+256); Print(50,80,'F'); readkey;
TextMode(BW40); Print(25,40,'G'); readkey;
TextMode(BW40+256); Print(50,40,'H'); readkey;
TextMode(OldMode);
writeln('Lastmode = ',LastMode,' TextAttr = ',TextAttr); readln;
end.
```

2. Điều khiển bàn phím

Unit Crt cung cấp các hàm để nhận biết các phím được ấn:

• **Hàm KeyPressed: boolean** cho giá trị True nếu vùng đệm bàn phím khác rỗng (tức là trước đó đã có phím được ấn mà máy chưa xử lý) và cho False nếu trái lại. Hàm này chỉ kiểm tra mà không làm thay đổi trạng thái vùng đệm bàn phím. Vòng lặp `repeat until keypressed;` sẽ dừng chương trình cho đến khi có một phím được ấn.

• **Hàm ReadKey: char** lấy ra ký tự đầu tiên trong vùng đệm bàn phím và trả về ký tự này. Thông thường vùng đệm là rỗng, do đó khi gặp lệnh ReadKey chương trình dừng lại chờ người dùng ấn một phím, hàm trả về ký tự vừa ấn, tuỳ theo giá trị này ta có thể quyết định sự làm việc tiếp theo của chương trình. Khi ấn một phím thông thường, máy sẽ gửi vào vùng đệm bàn phím một ký tự, ký tự này có mã ASCII tương ứng, ví dụ chữ A có mã 65, phím Enter có mã 13, phím ESC có mã 27. Khi ấn các phím đặc biệt và các tổ hợp phím máy gửi vào vùng đệm hai ký tự (ký tự đầu tiên luôn có mã bằng 0, ký tự thứ hai có mã khác 0), ví dụ khi ấn phím mũi tên lên máy gửi vào vùng đệm hai ký tự có mã là (0, 72), Ctrl+F1 là (0, 94). Để nhận diện các phím đặc biệt ta phải hai lần gọi hàm ReadKey để lấy hai ký tự và dùng ký tự thứ hai để nhận biết phím đã ấn. Chương trình ReadKey.pas ở dưới cho ta cách nhận biết tất cả các phím thông thường, các phím đặc biệt và các tổ hợp phím. Vòng lặp sau dùng để làm sạch vùng đệm bàn phím: `while keypressed do ch:=readkey;`

Chương trình **KeyPres.pas**: in các kí tự ngẫu nhiên từ A đến Z với màu chữ trên màu nền, vị trí trên màn hình đều là ngẫu nhiên cho tới khi ấn một phím thì kết thúc.

```
uses crt;
var x,y,c: byte;
begin clrscr; randomize;
repeat  x:= random(80)+1; y := random(25)+1;
c:= random(26)+1; c:=c+64;
textbackground(random(8)); textcolor(random(16)); gotoxy(x,y);
```

```
if (x<>80) or (y<>25) then write(chr(c)); delay(500);
until keypressed;
textattr:=7; clrscr;
end.
```

Chương trình **ReadKey.pas**: in ra mã của phím được ấn (cả phím đặc biệt).

```
uses crt;
var  ch: char;
begin clrscr; writeln('Nhan mot phim : '); ch := readkey;
if ch = chr(0) then  begin ch := readkey;
            writeln('Ban vua an phim dac biet co ma 0 va ',ord(ch));
                   end
else writeln('Ban vua an phim co ma ASCII : ',ord(ch));
readln;
end.
```

Chương trình **MuiTen.pas**: vẽ một hình chữ nhật nền trắng có độ rộng 14 và chiều cao 5 kí tự, dùng 4 phím mũi tên để di chuyển hình chữ nhật theo 4 chiều, ấn ESC kết thúc chương trình.

```
uses crt;
procedure ChuNhat(x1,y1,mau: byte);
var i,j: byte;
begin textcolor(mau);
for i:=0 to 4 do for j:=0 to 13 do
    begin gotoxy(x1+j,y1+i); write(chr(219)) end;
gotoxy(x1,y1);
end;
var x,y,k: byte; ch: char;

begin clrscr; x:=40; y:=12; ChuNhat(x,y,white);
repeat ch:= readkey;
if ch=#0 then
  begin ch:= readkey; k:= ord(ch);
    case k of
     72: begin ChuNhat(x,y,Black); dec(y); ChuNhat(x,y,white) end;
     80: begin ChuNhat(x,y,Black); inc(y); ChuNhat(x,y,white) end;
     77: begin ChuNhat(x,y,Black); inc(x); ChuNhat(x,y,white) end;
     75: begin ChuNhat(x,y,Black); dec(x); ChuNhat(x,y,white) end;
    end;
  end;
until ch = #27;
end.
```

Chương trình **CrtDemo.pas**: tạo một cửa sổ ngẫu nhiên trên màn hình, màu chữ và màu nền cũng ngẫu nhiên. Trong cửa sổ này có thể nhập các dòng văn bản, ấn Enter con trỏ xuống dòng, dùng 4 phím mũi tên để di chuyển con trỏ nhập, ấn phím Ins để chèn một dòng trắng, ấn phím Del để xoá dòng có con trỏ, ấn Alt+R để liên tục sinh các ký tự ngẫu nhiên trong cửa sổ. Gõ Alt+W để tạo một cửa sổ soạn thảo mới. Kết thúc chương trình: ấn Alt+X, Ctrl+C hay ESC.

```
uses Crt;
var  OrigMode,LastCol,LastRow:Word; Ch:Char; Done:Boolean;

procedure Initialize;
begin  CheckBreak:=False; OrigMode:=LastMode;
```

```
TextMode(Lo(LastMode)+Font8x8);
LastCol:=Lo(WindMax)+1; LastRow:=Hi(WindMax)+1;
GoToXY(1,LastRow); TextBackground(Black); TextColor(White);
Write(' Ins-InsLine  ','Del-DelLine  ',#27#24#25#26'-Cursor  ',
      'Alt-W-Window  ','Alt-R-Random  ','Esc-Exit');
Dec(LastRow,80 div LastCol); Randomize;
end;

procedure MakeWindow;
var X,Y,Width,Height: Word;
begin  Width:=Random(LastCol-2)+2; Height:=Random(LastRow-2)+2;
X:=Random(LastCol-Width)+1; Y:=Random(LastRow-Height)+1;
Window(X,Y,X+Width,Y+Height);
if OrigMode = Mono then
  begin TextBackground(White); TextColor(Black); ClrScr;
    Window(X+1,Y+1,X+Width-1,Y+Height-1);
    TextBackground(Black); TextColor(White); ClrScr;
  end
else
  begin TextBackground(Random(8)); TextColor(Random(7)+9) end;
ClrScr;
end;

procedure RandomText;
begin repeat Write(Chr(Random(256-32)+32)) until KeyPressed;
end;

begin Initialize; MakeWindow;  Done:=False;
repeat  Ch:=ReadKey;
  case Ch of
   #0: begin Ch:=ReadKey;
         case Ch of
          #17: MakeWindow;              { Alt-W }
          #19: RandomText;              { Alt-R }
          #45: Done:=True;              { Alt-X }
          #72: GotoXY(WhereX,WhereY-1); { Up }
          #75: GotoXY(WhereX-1,WhereY); { Left }
          #77: GotoXY(WhereX+1,WhereY); { Right }
          #80: GotoXY(WhereX,WhereY+1); { Down }
          #82: InsLine;                 { Ins }
          #83: DelLine;                 { Del }
         end;
      end;
   #3 : Done:=True;                  { Ctrl-C }
   #13: WriteLn;                     { Enter }
   #27: Done:=True;                  { Esc }
   else Write(Ch);
  end;
until Done; TextMode(OrigMode);
end.
```

Biến CheckBreak: boolean (thuộc unit Crt) dùng để điều khiển việc dừng chương trình. Khi CheckBreak = True ta có thể dừng chương trình bất kỳ lúc nào bằng cách ấn Ctrl+Break, khi CheckBreak = False thì bỏ khả năng này.

3. Tạo âm thanh

Các hàm và thủ tục để tạo âm thanh:

• **Thủ tục Sound(n: word)**: làm cho loa phát ra một âm thanh có tần số là n Hertz cho đến khi gặp thủ tục Nosound.

• **Thủ tục Nosound**: tắt âm thanh đã phát ra do thủ tục Sound.

• **Thủ tục Delay(m: word)**: dừng chương trình trong thời gian m mili giây.

Có thể dùng ba thủ tục này để tạo một nốt nhạc. Một nốt nhạc được đặc trưng bởi cao độ và trường độ, cao độ được thể hiện bởi thủ tục Sound(n), trường độ được thể hiện bởi thủ tục Delay(m). Ví dụ để tạo nốt sol trồn dùng bộ ba lệnh: `sound(392); delay(1600); nosound;`

Cao độ của các nốt nhạc ở các bát độ thấp, trung bình, cao và rất cao (đơn vị là Hertz), trường độ các nốt (đơn vị là mili giây) cho trong bảng sau:

Cao độ các nốt					**Trường độ**	
Đô	131	262	523	1047	Nốt tròn	1600
Đô #	139	277	554	1109	Nốt trắng	800
Rê	147	294	587	1175	Nốt đen	400
Rê #	156	311	622	1245	Nốt móc đơn	200
Mi	165	330	659	1319	Nốt móc kép	100
Pha	175	349	698	1397	Nốt móc ba	50
Pha #	185	370	740	1480		
Sol	196	392	784	1568		
Sol #	208	415	831	1661		
La	220	440	880	1760		
La #	233	466	932	1865		
Si	247	494	988	1976		

Chương trình **BaoDong.pas**: tạo hồi còi báo động cho đến khi ấn một phím.

```
uses crt;
begin clrscr; writeln('An phim bat ky de ket thuc bao dong');
repeat sound(359); delay(400); sound(200); delay(350)
until keypressed; nosound;
end.
```

Chương trình **LangToi.pas**: chơi bản nhạc "Làng tôi" của Văn Cao. Hệ số k dùng để điều khiển tốc độ chơi nhanh hay chậm của bản nhạc. Nếu nhập k>1 (ví dụ 1.5, 2.0) thì bản nhạc chơi chậm lại, nếu k<1 (ví dụ 0.8) bản nhạc chơi nhanh lên, tuỳ theo tốc độ máy mà nhập k cho thích hợp. Thủ tục `Tn(c,t)` nhằm tạo một nốt nhạc có cao độ là `c` hertz và trường độ là `k*t` mili giây.

```
uses crt;
var dem : integer; k : real; ch : char;
procedure tn(c: word; t: word);
      begin sound(c); delay(trunc(k*t)); nosound  end;

begin clrScr; dem := 0; writeln('LANG TOI cua Van Cao');
write('Vao he so toc do k = '); readln(k);
repeat inc(dem); Writeln('Choi ban nhac lan thu ',dem);
```

```
tn(262,800); tn(330,400); tn(392,800); tn(440,400); tn(392,800);
tn(392,400); tn(523,400); tn(494,400); tn(440,400); tn(392,800);
tn(440,400); tn(392,400); tn(349,400); tn(330,400); tn(392,1200);
tn(262,800); tn(294,400); tn(330,800); tn(392,400); tn(523,800);
tn(587,400); tn(659,800); tn(587,400); tn(523,400); tn(587,400);
tn(523,400); tn(392,400); tn(330,400); tn(392,400); tn(523,1200);
tn(523,800); tn(523,400); tn(440,800); tn(440,400); tn(494,800);
tn(494,400); tn(392,1200); tn(349,800); tn(349,400); tn(440,800);
tn(440,400); tn(330,800); tn(294,400); tn(392,1200); tn(262,800);
tn(294,400); tn(330,800); tn(349,400); tn(392,800); tn(330,400);
tn(294,800); tn(262,400); tn(523,800); tn(494,400); tn(587,800);
tn(392,400); tn(523,1200);
write('Choi nhac tiep khong (C/K)? '); ch:=readkey; writeln;
until upcase(ch) = 'K';
end.
```

Bài tập

1. Tạo các cửa sổ ngẫu nhiên trên màn hình với màu nền ngẫu nhiên, toạ độ cửa sổ ngẫu nhiên. Khi nào ấn phím bất kỳ thì kết thúc chương trình.

2. Lập một menu dọc để tính toán diện tích, chu vi , thể tích của các hình. Màn hình được bố trí như sau:

TINH TOAN HINH HOC
Lập trình: Bùi Thế Tâm

1. Tính diện tích hình chữ nhật
2. Tính diện tích hình thang
3. Tính chu vi đường tròn
4. Tính thể tích hình cầu
5. Kết thúc chương trình

HINH THANG
Vào đáy dài : 12.3
Vào đáy ngắn : 6.7
Vào chiều cao : 5.43
Dien tích là : 51.58500

Như vậy trên màn hình có ba cửa sổ. Cửa sổ trên ghi quảng cáo của chương trình. Cửa sổ giữa là các mục của menu và có một mục được chiếu sáng, để di chuyển hộp sáng trên menu ta dùng các phím mũi tên lên và xuống, muốn lựa chọn mục nào thì di chuyển hộp sáng đến mục đó và ấn Enter. Khi đã lựa chọn một mục thì cửa sổ thứ ba phía dưới màn hình xuất hiện, trong cửa sổ này ta tiến hành nhập dữ liệu cho bài toán và in ra kết quả tính.

3. Tạo một menu ngang nằm trên dòng thứ nhất của màn hình, khi chọn một mục của menu ngang lại hiện một menu dọc tương ứng. Nếu chọn một mục của menu dọc này thì một hàm được thực hiện. Hệ thống menu như vậy gọi là menu hai mức.

4. Lập trình cho phép dùng bàn phím của máy tính để chơi nhạc (các nốt nhạc được gắn với các phím tương ứng trên bàn phím), ví dụ ấn phím D sinh nốt đô, ấn phím R sinh nốt rê . . .

Chương 8

Con trỏ

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. Ví dụ khi khai báo var A: array[1..5000] of real; máy sẽ phân một vùng nhớ cố định 30000 byte cho mảng A, trong chương trình có thể ta không dùng đến cả 5000 phần tử. Để tránh lãng phí bộ nhớ Turbo Pascal cho phép dùng biến động (dynamic variable). Các biến này được lưu trữ trong vùng Heap (vùng nhớ tự do của máy), khi cần chúng có thể được tạo ra để chứa dữ liệu, khi không cần có thể xoá chúng đi để giải phóng bộ nhớ. Biến động không có tên và do một con trỏ quản lý.

1. Khai báo kiểu con trỏ và biến con trỏ

Kiểu con trỏ là một kiểu dữ liệu đặc biệt dùng để lưu trữ những giá trị địa chỉ. Biến con trỏ có độ lớn 4 byte : hai byte thấp chứa địa chỉ offset, 2 byte cao chứa địa chỉ segment. Nhờ biến con trỏ ta có thể xin cấp phát động trên vùng Heap. Có hai kiểu con trỏ: con trỏ định kiểu và con trỏ không định kiểu.

Con trỏ định kiểu là con trỏ cần xác định kiểu dữ liệu mà nó trỏ đến. Cách khai báo kiểu con trỏ và biến con trỏ (theo hai cách):

```
Type  KiểuConTrỏ = ^KiểuDữLiệu ;
Var   BiếnConTrỏ1 : KiểuConTrỏ ;  BiếnConTrỏ2 : ^KiểuDữLiệu ;
```

trong đó KiểuDữLiệu có thể là real, integer, char, mảng, bản ghi . . . Khi một biến con trỏ chứa địa chỉ byte đầu tiên của một vùng nhớ thì ta nói con trỏ đang trỏ tới vùng nhớ này. Con trỏ real trỏ tới vùng nhớ 6 byte, con trỏ integer trỏ tới vùng nhớ 2 byte. . . Kiểu con trỏ cũng có thể là thành phần của bất kỳ một kiểu có cấu trúc nào (mảng, bản ghi). Hai con trỏ định kiểu gọi là tương thích (trong các phép gán, so sánh, truyền dữ liệu cho tham số) nếu cùng trỏ tới một kiểu dữ liệu.

Con trỏ không định kiểu là con trỏ không quan tâm đến kiểu dữ liệu mà nó đang trỏ tới, chỉ quan tâm đến địa chỉ. Cách khai báo: `Var P: Pointer ;` Con trỏ không định kiểu tương thích với mọi kiểu con trỏ khác.

Ví dụ:

```
Type  IntPtr = ^Integer ;
      SVienPtr = ^SVien ; { Kiểu bản ghi SVien có thể khai sau }
      SVien = Record
                HoTen : string[25] ;
                DiemTB : real ;
              End ;
Var  P1 : IntPtr ; P : SVienPtr ;  P2: ^Word ; Q: Pointer ;
```

2. Các thao tác đối với con trỏ

Gán một giá trị địa chỉ cho con trỏ theo các cách: `P := @x ; P := Addr(x) ; P := Ptr(segment, offset) ;` trong đó P là biến con trỏ bất kỳ, x là tên biến (hoặc tên hàm, thủ tục). Các hàm Addr, Ptr và toán tử @ trả về một địa chỉ kiểu Pointer.

Hằng NIL. Lệnh gán P := NIL nhằm làm cho con trỏ P không trỏ vào đâu cả (P là con trỏ kiểu bất kỳ).

Phép gán (:=) giữa hai con trỏ: hai con trỏ có thể gán cho nhau trong trường hợp tương thích, khi đó chúng cùng trỏ tới một địa chỉ. Hai con trỏ định kiểu trỏ tới các kiểu dữ liệu khác nhau là không tương thích, song chúng lại tương thích với kiểu Pointer. Ví dụ sau khai báo `Var p, s: byte; q : ^real; r: pointer ;` thì lệnh gán `p := s` là đúng, lệnh gán `q := p ;` là sai, song hai lệnh sau là đúng : `r := p ; q := r ;`

Hai giá trị con trỏ có thể so sánh bằng nhau (=) hay khác nhau (<>) nếu chúng tương thích. Với các giá trị con trỏ không có các phép so sánh > >= < <=.

Để truy cập nội dung vùng dữ liệu mà con trỏ P đang trỏ ta dùng ký hiệu P^. P^ xem như là một biến thông thường. Nếu P là con trỏ định kiểu thì P^ là biến có kiểu là kiểu dữ liệu mà P trỏ tới. Nếu P là con trỏ không định kiểu thì P^ là một biến không định kiểu có địa chỉ lưu trong P, nó được dùng khi không quan tâm đến kiểu dữ liệu được truy cập.

Chương trình **ConTro.pas** minh hoạ ý nghĩa của biến con trỏ:

```
var x: real; p,q: ^real;
begin x:= 123.456789; p:= @x; q:= addr(x);
writeln('x = ',x:13:5,'  p^ = ',p^:13:5,'  q^ = ',q^:13:5);
writeln('Dia chi bien x : ',seg(x),':',ofs(x));
writeln('Dia chi luu trong con tro p : ',seg(p^),':',ofs(p^));
readln; end.
```

3. Cấp phát động

Đối với biến con trỏ ta có thể dùng kỹ thuật cấp phát động nhờ các thủ tục New, Getmem để tạo các biến động P^ mới. Vùng cấp phát động bao giờ cũng là vùng nhớ tự do (Heap). Turbo Pascal quản lý vùng Heap thông qua con trỏ Heap. Con trỏ Heap luôn luôn trỏ vào byte tự do đầu tiên của vùng nhớ còn tự do của Heap. Mỗi lần dùng thủ tục New tạo ra một biến động thì con trỏ Heap dịch chuyển về phía đỉnh của vùng nhớ tự do một số byte tương ứng với kích thước của biến động mới được tạo.

Các hàm và thủ tục liên quan tới cấp phát động:

• **Thủ tục New(P)**, trong đó P là một con trỏ định kiểu, sẽ cấp phát một vùng nhớ trên Heap có kiểu và kích thước do P quy định, P trỏ tới vùng nhớ vừa cấp (tức là P chứa địa chỉ byte đầu tiên của vùng nhớ), tạo ra biến định kiểu P^ (gọi là biến động) và có thể dùng P^ như một biến thông thường. Vùng nhớ đã cấp phát được hệ thống bảo vệ cho đến khi bị thu hồi. Nếu dùng New(P) nhiều lần thì P chứa địa chỉ vùng nhớ được cấp phát lần cuối cùng.

• **Thủ tục Dispose(P)** thu hồi vùng nhớ đã được cấp phát cho con trỏ P bởi New(P)

Chương trình **SVien.pas** minh hoạ cách dùng các thủ tục New và Dispose:

```
type SVienPtr = ^SVien;
     SVien = Record Hoten: string[25]; DiemTb: real end;
var p,q: SVienPtr;
begin  new(p); p^.hoten:= 'Bui The Tam'; p^.diemtb:= 5.5;
       q:= p; new(p);
write('Vao ho ten : '); readln(p^.hoten);
write('Vao diem trung binh : '); readln(p^.diemtb);
writeln('Sinh vien 1: ',q^.hoten,'  ',q^.diemtb:5:2);
writeln('Sinh vien 2: ',p^.hoten,'  ',p^.diemtb:5:2);
dispose(q); dispose(p);
readln; end.
```

Chương trình **CTMang.pas** nêu cách khai báo một biến con trỏ kiểu mảng và cách truy nhập các phần tử của mảng.

```
const nmax = 1000;
type mang= array[1..nmax] of integer;
var a: ^mang; i,n, tong: integer;
begin write('Vao n = '); readln(n); new(a); tong := 0;
for i:=1 to n do
 begin write('a',i,' = '); readln(a^[i]); tong:= tong+a^[i] end;
write('Mang A: '); for i:=1 to n do  write(a^[i],'   ') ; writeln;
dispose(a); writeln('Tong = ',tong); readln;
end.
```

• **Thủ tục GetMem(var P : pointer; n : word)** cấp phát n byte trên Heap cho con trỏ P mà không quan tâm đến kiểu dữ liệu, P chứa địa chỉ byte đầu tiên của vùng nhớ, biến động P^ được dùng như một biến không định kiểu. Khối nhớ lớn nhất có thể tạo trên Heap là 65528 byte.

• **Thủ tục FreeMem(var P : pointer; n : word)** thu hồi vùng nhớ n byte được cấp phát bằng GetMem cho con trỏ P.

Chương trình **MangDong.pas** minh hoạ cách dùng các thủ tục Getmem, Freemem để tạo các mảng động (kích thước của chúng không cần ấn định trước) nhằm tiết kiệm bộ nhớ. Lúc đầu mảng chỉ khai tượng trưng có 1 phần tử, sau đó mảng được cấp phát vùng nhớ vừa đủ cho n phần tử cần dùng.

```
type mang=array[1..1] of real;
var a:^mang; i,n: integer; tong: real;
begin write('n = '); readln(n); tong:= 0;
Getmem(a,n* sizeof(real));
for i:=1 to n do
  begin write('a',i,' = '); readln(a^[i]); tong:= tong+a^[i] end;
for i:=1 to n do  writeln('a[',i,'] = ',a^[i]:13:5);
Freemem(a,n* sizeof(real)); writeln('Tong = ',tong:13:5); readln;
end.
```

• **Biến HeapEnd: pointer** trỏ tới điểm cuối của Heap được chương trình sử dụng, **biến HeapOrg: pointer** trỏ tới điểm bắt đầu của Heap được dùng cấp phát động. Các biến HeapEnd và HeapOrg được hệ thống khởi gán khi chương trình bắt đầu chạy và không thay đổi khi chương trình chạy.

• **Biến HeapPtr: pointer** trỏ tới đỉnh của Heap, tức là đỉnh của vùng nhớ cấp

phát động và cũng là đáy của vùng nhớ tự do. Khởi đầu giá trị HeapPtr = HeapOrg.

• **Hàm MemAvail: LongInt** cho tổng số byte còn tự do trên Heap. **Hàm MaxAvail: LongInt** cho số byte liên tục lớn nhất còn tự do trên Heap. Các vùng nhớ còn tự do trên Heap thường bị phân ra thành các khối nhỏ do máy cấp phát và giải toả các vùng nhớ trên Heap không theo một thứ tự nào. Để tránh tràn Heap, trước khi cấp phát bộ nhớ cho một đối tượng có kích thước lớn (như mảng, mảng các bản ghi) ta nên dùng hàm MaxAvail để kiểm tra có đủ bộ nhớ không.

Chương trình **Heap.pas** minh hoạ cách dùng các biến và hàm ở trên để quản lý Heap. Trong Turbo Pascal cho phép khai báo một mảng có kích thước không quá 64 KB.

```
type mang1 = array[1..65535] of byte;
     mang2 = array[1..100, 1..109] of real;
var a: ^mang1; b: ^mang2; top: pointer; i,j: longint;
begin writeln('HeapOrg = ',seg(HeapOrg^),':',ofs(HeapOrg^));
writeln('HeapEnd = ',seg(HeapEnd^),':',ofs(HeapEnd^));
writeln('HeapPtr = ',seg(HeapPtr^),':',ofs(HeapPtr^));
mark(top);
new(a); for i:=1 to 65535 do a^[i]:=1;
writeln('MemAvai = ',memavail,'  MaxAvail = ',maxavail);
if MaxAvail >= sizeof(mang2) then
  begin new(b);
     for i:=1 to 100 do for j:=1 to 109 do b^[i,j]:=1.234;
     writeln('MemAvai = ',memavail,'  MaxAvail = ',maxavail);
  end;
writeln(a^[65535],'  ',b^[100,109]:8:3); release(top);
writeln('MemAvai = ',memavail,'  MaxAvail = ',maxavail);
readln ; end.
```

• **Thủ tục Mark(var Top: pointer)** gán giá trị của con trỏ Heap cho con trỏ Top. Thủ tục này dùng để đánh dấu địa chỉ bắt đầu cấp phát động.

• **Thủ tục Release(var Top: pointer)**: thu hồi tất cả các vùng nhớ được cấp phát bắt đầu từ địa chỉ được lưu trong con trỏ Top nhờ thủ tục Mark(Top) cho đến đỉnh Heap hiện tại. Thủ tục Mark và Release nhằm xoá một loạt các biến động khỏi Heap.

Chương trình **MangCT.pas** minh hoạ cách dùng các thủ tục Mark, Release và mảng các con trỏ.

```
const nmax = 1000;
var b: array[1..nmax] of ^integer; i,n,tong: integer; top: pointer;
begin write('Vao n = '); readln(n); tong:=0; mark(top);
for i:=1 to n do  begin new(b[i]);
     write('b',i,' = '); readln(b[i]^); tong:= tong+b[i]^; end;
for i:=1 to n do  write(b[i]^,'   ') ; writeln; release(top);
writeln('Tong = ',tong); readln;
end.
```

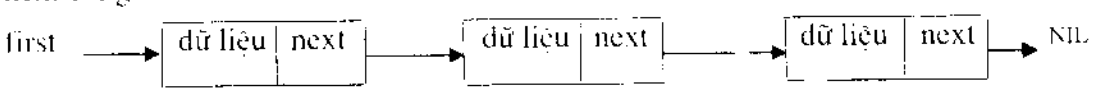
4. Danh sách liên kết

a) Danh sách liên kết thuận

Danh sách là một dãy hữu hạn các phần tử thuộc cùng một lớp đối tượng nào đó. Ví dụ : danh sách sinh viên, danh sách vật tư, danh sách các hoá đơn, danh sách các số

thực. Trong các bài trước ta đã **dùng mảng** để biểu thị một danh sách. Cách này có các nhược điểm: kích thước của mảng phải định trước nên tốn bộ nhớ (số phần tử thực tế dùng nhiều khi rất ít so với khai báo), khi thêm một phần tử vào mảng hoặc xoá một phần tử ra khỏi mảng ta phải mất nhiều thời gian để dồn mảng.

Danh sách liên kết dùng để cài đặt một danh sách sẽ khắc phục được các nhược điểm trên của mảng. Danh sách liên kết thuận gồm nhiều phần tử nằm không liên tục trong Heap. Cấu tạo của danh sách liên kết thuận: có một con trỏ **first** chứa địa chỉ của phần tử đầu tiên của danh sách, phần tử đầu có phần dữ liệu và một con trỏ **next** để chứa địa chỉ của phần tử thứ hai, tổng quát phần tử thứ i có phần dữ liệu và một con trỏ next để chứa địa chỉ của phần tử thứ i+1, đối với phần tử cuối cùng giá trị của con trỏ next bằng NIL.



Để thuận tiện khi thêm phần tử mới vào cuối danh sách liên kết ta dùng một con trỏ **last** chứa địa chỉ của phần tử cuối cùng. Khởi tạo một danh sách rỗng first = NIL.

Chương trình **Dslk.pas** minh hoạ các cách làm việc với danh sách liên kết thuận. Phần dữ liệu của một phần tử là các thông tin về một sinh viên. Chương trình có 8 chức năng:

1. Bổ sung các sinh viên mới vào cuối danh sách liên kết (dùng thủ tục Bosung).

2. Duyệt danh sách liên kết từ phần tử đầu đến phần tử cuối (thủ tục Duyet).

3. Vào một mã sinh viên từ bàn phím, tìm trong danh sách sinh viên có mã vừa nhập, chèn một phần tử mới vào sau phần tử vừa tìm thấy (thủ tục ChenSau).

4. Vào một mã sinh viên từ bàn phím, tìm trong danh sách sinh viên có mã vừa nhập, xoá phần tử vừa tìm thấy (thủ tục TimXoa).

5. Sắp xếp danh sách theo trường Maso tăng dần (thủ tục SapXep). Khi sắp xếp ta chỉ cần trao đổi phần dữ liệu của phần tử, phần con trỏ giữ nguyên.

6. Ghi danh sách liên kết đã tạo vào tệp các bản ghi SVien.dat (thủ tục GhiTep).

7. Tạo danh sách liên kết từ tệp các bản ghi SVien.dat có sẵn (thủ tục DocTep). Sau khi tạo ta có thể bổ sung theo chức năng 1.

8. Xoá toàn bộ danh sách liên kết khỏi Heap và thoát khỏi chương trình.

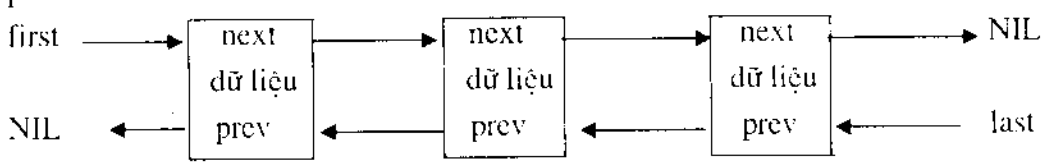
```
uses crt;
type nodePtr = ^node;
     svien = record maso: string[6];
                    hoten: string[23];
                    dtb: real
             end;
     node = record  data: svien;
                    next : nodePtr
            end;
var  first, last : nodePtr; chon: byte; traloi: char;

procedure Bosung;
var  p: nodePtr; ans: integer;
begin
```

```
while true do
 begin  new(p);
     with p^.data do begin
       write('Ma sinh vien : '); readln(maso);
       write('Ho va ten : '); readln(Hoten);
       write('Diem trung binh: '); readln(dtb);
                        end;
     p^.next:=NIL;
     if first=NIL then begin first:= p; last:= p end
                  else begin last^.next:= p; last:= p end;
     write('Co tiep tuc khong 1/0 ? ');  readln(ans);
     if ans=0 then break;
 end;
end;

procedure DuyetXuoi;
var  p: nodePtr;
begin if first = NIL then begin writeln('D.sach rong'); exit end;
p := first;
while p <> NIL do
  begin  with p^.data do writeln(maso:5,'  ',Hoten:25,'  ',dtb:5:2);
         p := p^.Next;
  end;
end;

procedure ChenSau;
var q,p: nodePtr; found: boolean; masv: string[6];
begin  if first <> NIL then  begin
write('Ma so SV can loai khoi danh sach : '); readln(masv);
p:= first; found:= false;
while (p<>NIL) and (not found) do
   if p^.data.maso=masv then found := true else p:=p^.next;
if found then
  begin  new(q);
         with q^.data do begin
                  write(' Ma so    : '); readln(maso);
                  write(' Ho ten   : '); readln(Hoten);
                  write(' Diem trung binh : '); readln(dtb);
                        end;
         q^.next:= p^.next; p^.next:= q;
  end else begin write('Khong tim thay...'); readln end;
                              end;
end;

procedure TimXoa;
var masv: string[6]; found: boolean; p,q: nodePtr;
begin write('Ma so SV can loai khoi danh sach : '); readln(masv);
p:= first;
if p<>NIL then  begin   found:= false;
  while (p<>NIL) and (not found) do
    if p^.data.maso=masv then found := true
                         else begin q:=p; p:=p^.next end;
  if found then
     begin if p=first then first :=p^.next else q^.next:=p^.next;
           if p^.next=NIL then last:=q; dispose(p)
     end else begin write('Khong tim thay...'); readln end;
     •
```

```
                end;
end;

procedure SapXep;
var p, q, min: nodePtr; x: svien;
begin  p:= first;
while p<>last do
 begin q:= p^.next; min:= p;
       while q<> NIL do
         begin  if q^.data.maso < p^.data.maso then min:= q;
                q:= q^.next
         end;
       x:= p^.data; p^.data:= min^.data; min^.data:= x;
       p:= p^.next;
 end;  write('Sap xep xong .... '); readln;
end;

procedure GhiTep;
var x: svien; f: file of svien; p: nodePtr;
begin assign(f,'Svien.dat'); rewrite(f);
p:= first;
while p<>NIL do begin x:= p^.data; write(f,x); p:= p^.next end;
close(f); write('Ghi xong ... '); readln;
end;

procedure DocTep;
var x: svien; f: file of svien; p: nodePtr;
begin assign(f,'Svien.dat'); reset(f);
while not eof(f) do
 begin  read(f,x); new(p); p^.data := x;  p^.next:=NIL;
     if first=NIL then begin first:= p; last:= p end
                  else begin last^.next:= p; last:= p end;
 end;
close(f); write('Doc xong ... '); readln;
end;

Begin clrscr; First := NIL;
repeat
writeln; writeln('1. Bo sung mot sinh vien');
writeln('2. Duyet danh sach sinh vien');
writeln('3. Tim kiem mot phan tu va chen vao sau');
writeln('4. Tim kiem mot phan tu va xoa');
writeln('5. Sap xep danh sach theo Maso tang dan');
writeln('6. Ghi danh sach vao tep cac ban ghi');
writeln('7. Tao danh sach lien ket tu tep');
writeln('8. Ket thuc chuong trinh');
writeln; write('Chon chuc nang: '); readln(chon); writeln;
case chon of
 1: Bosung;    2: DuyetXuoi;    3: ChenSau;
 4: TimXoa;     5: SapXep;       6: GhiTep;   7: DocTep;
end;
until chon=8;
while first<>NIL do begin last:= first; first:= first^.next;
  ;                       dispose(last) end;
End.
```

b) Danh sách liên kết kép

Danh sách liên kết kép là danh sách mà mỗi phần tử của nó gồm ba thành phần: phần dữ liệu, con trỏ **next** chứa địa chỉ phần tử tiếp sau, con trỏ **prev** chứa địa chỉ của phần tử liền trước, con trỏ next của phần tử cuối cùng trong danh sách bằng NIL, con trỏ prev của phần tử đầu tiên cũng bằng NIL. Danh sách liên kết kép hoàn toàn xác định bởi hai con trỏ: con trỏ **first** chứa địa chỉ phần tử đầu tiên, con trỏ **last** chứa địa chỉ phần tử cuối cùng. Danh sách liên kết kép cho phép dễ dàng xác định phần tử đứng trước một phần tử đã biết, do đó nó thuận tiện hơn danh sách liên kết thuận trong các thao tác: duyệt ngược danh sách, chèn một phần tử mới vào trước một phần tử, xoá một phần tử.



Chương trình **DslkKep.pas** minh hoạ cách làm việc với danh sách liên kết kép. Chương trình có 6 chức năng:

1. Bổ sung các phần tử mới vào cuối danh sách (thủ tục Bosung).

2. Duyệt danh sách theo chiều xuôi: từ phần tử đầu đến phần tử cuối (thủ tục DuyetXuoi).

3. Duyệt danh sách theo chiều ngược: từ phần cuối lên phần tử đầu tiên (thủ tục DuyetNguoc).

4. Vào một mã sinh viên từ bàn phím, tìm sinh viên trong danh sách có mã trùng với mã vừa nhập, chèn vào trước phần tử vừa tìm thấy một phần tử mới (thủ tục ChenTruoc)

5. Tìm một phần tử theo mã và xoá phần tử vừa tìm thấy khỏi danh sách (thủ tục TimXoa).

6. Xoá toàn bộ danh sách và thoát khỏi chương trình.

```
uses crt;
type nodePtr = ^node;
 svien = record  maso : string[6];
                 hoten: string[23];
                 dtb  : real;
          end;
 node = record data : svien;
               prev, next : nodePtr;
        end;
var first,last: nodePtr; chon: byte;

procedure Bosung;
var p,q: nodePtr; ans: char;
begin
while true do begin
  new(p); with p^.data do
            begin write('- Ma so: '); readln(maso);
                  write('- Ho va ten: '); readln(hoten);
```

```
                write('- Diem trung binh: '); readln(dtb); end;
  if first=NIL then  begin p^.next:= NIL;
                           p^.prev := NIL; first:= p; last:= p;  end
  else begin q:= last; q^.next:= p; p^.next:= NIL;
            p^.prev:= q; last:= p; end;
  write('Co tiep tuc khong C/K ?');  readln(ans);
  if upcase(ans) = 'K' then break;
            end;
end;

procedure DuyetXuoi;
var p: nodePtr;
begin
if first <> NIL then
   begin p:= first;
         while p<> NIL do
              begin with p^.data do
                    writeln(maso,'  ',hoten,'  ',dtb:1:2);
                    p:= p^.next;
              end;
   end
else writeln('Danh sach rong ...'); readln;
end;

procedure DuyetNguoc;
var p: nodePtr;
begin
if first <> NIL then
  begin p:= last;
        while p<> NIL do
          begin  with p^.data do
                 writeln(maso,'  ',hoten,'  ',dtb:1:2);
                 p:= p^.prev;
          end;
  end
else writeln('Danh sach rong ...'); readln;
end;

procedure ChenTruoc;
var  p,q,r: nodePtr; found: boolean; masv: string[6];
begin write('Ma so SV can tim : '); readln(masv);
if first<>NIL then Begin
p:= first; found:= false;
while (p<>NIL) and (not found) do
    if p^.data.maso=masv then found := true else p:=p^.next;
if found then
  begin new(q);
        with q^.data do
            begin write('- Ma so: '); readln(maso);
                  write('- Ho va ten: '); readln(hoten);
                  write('- Diem trung binh: '); readln(dtb);
            end;
        if p=first then
           begin q^.next:= p; first^.prev:= q;  first:= q end
        else begin r:=p^.prev; q^.next:= r^.next; r^.next:= q;
                  q^.prev:= p^.prev; p^.prev:= q end;
```

```
  end
else begin write('Khong tim thay...'); readln end;
                  End;
end;

procedure TimXoa;
var  p,q,pt: nodePtr; found: boolean; masv: string[6];
begin
write('Ma so SV can loai khoi danh sach : '); readln(masv);
if first<>NIL then begin
p:= first; found:= false;
while (p<>NIL) and (not found) do
      if p^.data.maso=masv then found := true else p:=p^.next;
if found then
  begin  q:= p^.prev; pt:= p^.next; if q<>NIL then q^.next:= p^.next;
    if pt<>NIL then pt^.prev:= p^.prev;
    if p=first then first:= pt; if p=last then last:= q;
    dispose(p);
  end
else begin  write('Khong tim thay ...'); readln end;
                   end;
end;

Begin first:= NIL; clrscr;
repeat  writeln('1. Bo sung phan tu');
  writeln('2. Duyet danh sach theo chieu xuoi');
  writeln('3. Duyet danh sach theo chieu nguoc');
  writeln('4. Chen vao truoc mot phan tu');
  writeln('5. Tim kiem va xoa mot phan tu');
  writeln('6. Ket thuc chuong trinh');
  write('   Chon chuc nang : '); readln(chon);
  case chon of
   1: Bosung;     2: DuyetXuoi;   3: DuyetNguoc;
   4: ChenTruoc;  5: TimXoa;
  end;
until chon=6;
while first<> NIL do begin last:=first; first:=first^.next;
                           dispose(last); end;
End.
```

c) Các dạng danh sách liên kết thường dùng khác

• **Danh sách liên kết ngược**: mỗi phần tử gồm có thành phần dữ liệu và một con trỏ prev chứa địa chỉ phần tử liền trước, con trỏ prev của phần tử đầu tiên bằng NIL. Danh sách xác định bởi con trỏ last chứa địa chỉ phần tử cuối. Danh sách liên kết ngược tiện lợi khi duyệt qua danh sách theo chiều ngược từ cuối lên đầu danh sách. Khai báo dữ liệu:

```
Type nodePtr = ^node;
     node = record
                data : integer;
                prev : nodePtr;
              end;
Var last : nodePtr;
```

• **Danh sách liên kết vòng tròn** là danh sách liên kết thuận nhưng con trỏ next của phần tử cuối cùng không chứa NIL mà lại chứa địa chỉ của phần tử đầu. Trong

danh sách này mọi phần tử đều bình đẳng, đều có phần tử đi sau. Xuất phát từ một phần tử bất kỳ ta có thể truy cập đến phần tử bất kỳ trong danh sách.

• **Danh sách liên kết vòng tròn có đầu** là danh sách liên kết vòng tròn mà trong đó có một phần tử gọi là đầu danh sách. Đầu danh sách chứa một giá trị đặc biệt (tự quy định) để phân biệt với các phần tử khác của danh sách (ví dụ nếu danh sách là các số dương thì phần tử đầu có thể lấy giá trị bằng -1). Ưu điểm của danh sách liên kết vòng tròn có đầu là nó không bao giờ rỗng.

• **Danh sách liên kết kép vòng tròn có đầu** là danh sách liên kết kép mà con trỏ next của phần tử cuối cùng chứa địa chỉ phần tử đầu, con trỏ prev của phần tử đầu chứa địa chỉ phần tử cuối cùng, và có một phần tử là đầu danh sách (có dấu hiệu để khác với tất cả các phần tử khác). Trong các ứng dụng người ta hay dùng loại danh sách này.

5. Kiểu dữ liệu ngăn xếp - stack

Stack là một danh sách theo đó tất cả các công việc chèn và huỷ đều được thực hiện ở một đầu của danh sách (gọi là đỉnh của ngăn xếp). Stack giống như một chồng đĩa, đĩa nào đặt cuối cùng lên đỉnh chồng thì đĩa đó sẽ được lấy ra đầu tiên. Do đó Stack còn có tên gọi là LIFO (last in first out - vào sau ra trước). Việc thêm một phần tử vào stack có tên gọi là đẩy (Push) vào stack, còn việc huỷ một phần tử khỏi stack gọi là lấy (Pop) khỏi stack.

a) Stack dựa trên mảng. Ta dùng một mảng để chứa các phần tử của stack. Song ta cần một biến đếm Top để biết đỉnh của stack đang ở phần tử thứ mấy của mảng, lúc đầu Top=0 (stack rỗng). Biến Top cũng cho biết tổng số phần tử của stack.

Chương trình **StMang.C** minh hoạ cách làm việc với stack. Đầu chương trình khai báo một biến cấu trúc s có hai thành phần: mảng **a** lưu các phần tử của stack là các số nguyên dương, biến Top để nhớ đỉnh của stack. Thủ tục Push(n) để đẩy một số nguyên vào stack. Hàm Pop() để lấy số nguyên ở đỉnh stack. Thủ tục Duyet() để xem nội dung hiện tại của stack.

```
uses crt;
const max=100;
type stack = record
               a: array[1..max] of integer;
               top: 0..max ;
             end;
var s: stack; n,chon: integer;

procedure Push(n: integer);
begin  if s.top>=max then writeln('Ngan xep day')
       else begin  inc(s.top); s.a[s.top]:= n end;
end;

function Pop : integer;
var n: integer;
begin
if s.top=0 then begin writeln('Stack rong'); pop:= 0 end
else begin n:= s.a[s.top]; dec(s.top); pop:=n end;
end;

procedure Duyet;
var i: integer;
```

```
begin if s.top=0 then writeln('Stack rong')
else for i:=1 to s.top do writeln('Phan tu ',i,' = ',s.a[i]);
end;

Begin  clrscr; s.top:=0;
repeat write('1. Push  2. Pop  3. Duyet  4. Thoat. Chon: ');
read(chon);
case chon of
1: begin write('Vao n : '); readln(n); Push(n) end;
2: begin n:= Pop; if n<>0 then writeln('Phan tu lay ra = ',n) end;
3: Duyet;
end;
until chon=4;
end.
```

b) Stack dùng danh sách liên kết. Stack dùng danh sách liên kết hoàn toàn giống danh sách liên kết thuận, nhưng chỉ có điều khác là khi thêm phần tử mới hay huỷ một phần tử ta luôn luôn làm ở đầu danh sách. Do đó ta phải duy trì một con trỏ Top để trỏ vào phần tử đầu tiên của danh sách (đỉnh của stack).

Chương trình **StDslk.pas** minh hoạ cách làm việc với stack dùng danh sách liên kết, các phần tử của stack là các số nguyên dương.

```
uses crt;
type  StackPtr= ^node;
      node = record
               data: integer;
               next: StackPtr;
             end;
var top: StackPtr; chon,n: integer;

procedure Push(n: integer);
var p: StackPtr;
begin new(p); p^.data:= n; p^.next:= top; top:= p;
end;

function Pop: integer;
var p: StackPtr;
begin  if top=NIL then pop:=0
       else begin pop:= top^.data; p:= top;
                  top:= top^.next;  dispose(p)  end;
end;

procedure Duyet;
var p: StackPtr;
begin
if top<>NIL then
  begin p:=top;
       while p<>NIL do begin writeln(p^.data,'  '); p:=p^.next end;
  end
else writeln('Stack rong'); readln;
end;

Begin clrscr; top:=NIL;
repeat write('1. Push  2. Pop  3. Duyet  4. Thoat. Chon: ');
read(chon);
case chon of
```

```
1: begin write('Vao n : '); readln(n); Push(n) end;
2: begin n:= Pop; if n<>0 then writeln('Phan tu lay ra = ',n)
                  else writeln('Stack rong');
   end;
3: Duyet;
end;
until chon=4;
end.
```

6. Kiểu dữ liệu hàng đợi - queue

Queue là một danh sách mà việc thêm một phần tử mới được thực hiện ở đuôi queue, việc huỷ một phần tử được thực hiện ở đầu queue. Queue giống như một dãy khách hàng xếp hàng trả tiền trong siêu thị, người xếp hàng trước sẽ được phục vụ trước và ra khỏi hàng, người mới tham gia xếp hàng thì xếp vào cuối hàng. Do đó queue còn có tên gọi FIFO (first in first out - vào trước thì ra trước).

a) Queue dựa trên mảng

Ta dùng một mảng **a**[1..max] để chứa các phần tử của queue. Song ta cần 3 biến nguyên: **front** để đánh dấu phần tử đầu tiên của queue đang là phần tử thứ mấy của a, biến nguyên **back** để đánh dấu phần tử cuối cùng của queue đang là phần tử thứ mấy của mảng a, biến **count** để đếm số phần tử hiện tại của queue. Lần đầu tiên đưa vào queue một phần tử thì ta xếp nó vào a[1] và cho front = 1, back = 1, count = 1. Tiếp theo phần tử thứ hai được xếp vào a[2] và back = 2. . . Nếu tổng số phần tử của queue bằng max thì queue bị đầy, ta chỉ có thể huỷ bớt phần tử. Mỗi lần huỷ một phần tử của queue thì biến front tăng lên 1.

Giả sử back = max (a[max] đã có phần tử của queue), các phần tử ở đầu mảng a còn trống và ta muốn thêm một phần tử mới x vào queue. Khi đó ta có thể xếp phần tử x này vào a[1] và cho back = 1. Queue có khả năng này gọi là queue vòng tròn.

Chương trình **QueMang.pas** tạo một queue vòng tròn chứa các số nguyên dương. Thủ tục Putq(n) đặt một phần tử mới vào queue. Hàm Getq để lấy một phần tử ra khỏi queue. Thủ tục Duyet: xem nội dung của queue.

```
uses crt;
const max = 5;
type queue= Record
              a: array[1..max] of integer;
              front, back, count : integer;
            end;
var q: queue; n,c: integer;

procedure putq(n: integer);
begin
if q.count=0 then
   begin q.front:=1; q.back:=1; q.count:=1; q.a[1]:=n end
else begin q.back:= (q.back mod max) + 1;
           q.a[q.back]:= n; inc(q.count)
     end;
end;

function getq: integer;
begin
```

```
dec(q.count); getq:= q.a[q.front]; q.a[q.front] := 0;
q.front := (q.front mod max) +1;
if q.count=0 then begin q.front:=0; q.back:=0 end;
end;

procedure Duyet;
var i: integer;
begin
if q.count<>0 then
  begin
    writeln('front= ',q.front,' back= ',q.back,' count= ',q.count);
    for i:=1 to max do write(q.a[i],'  '); writeln;
  end else  writeln('Queue rong')
end;

Begin clrscr; q.front:=0; q.back:=0; q.count:= 0;
      for c:=1 to max do q.a[c]:=0;
repeat
write('1. Them, 2. Lay ra  3. Duyet  4. Ket thuc. Chon: ');
readln(c);
case c of
1: if q.count=max then writeln('Queue day')
   else begin write('Vao n = '); readln(n); putq(n) end;
2: if q.count=0 then  writeln('Queue rong')
   else  writeln('Da lay phan tu: ',getq);
3: Duyet;
end;
until c=4;
end.
```

b) Queue dùng danh sách liên kết. Queue dùng danh sách liên kết hoàn toàn giống danh sách liên kết thuận , nhưng chỉ có điều khác là khi thêm phần tử mới ta luôn luôn nối vào cuối danh sách, khi huỷ một phần tử ta luôn huỷ phần tử đầu tiên trong danh sách. Do đó ta phải duy trì hai con trỏ: front trỏ vào phần tử đầu, back trỏ vào phần tử cuối.

Chương trình **QueDslk.pas** tạo một hàng đợi gồm các số nguyên dương dùng danh sách liên kết thuận.

```
uses crt;
type  queuePtr= ^node;
      node = record
               data: integer;
               next: queuePtr;
             end;
var front,back: queuePtr; chon,n: integer;

procedure ThemVao(n: integer);
var p: queuePtr;
begin new(p); p^.data:= n; p^.next:= NIL;
if front=Nil then begin front:=p; back:=p end
else begin back^.next:=p; back:=p end;
end;

function LayRa: integer;
var p: queuePtr;
begin  if front=NIL then LayRa:=0
```

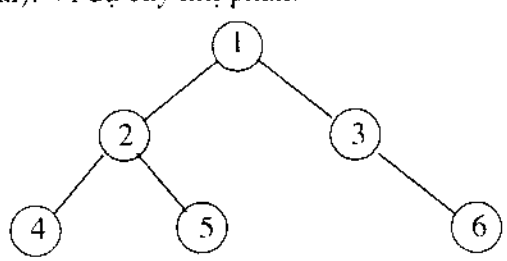
```
        else begin LayRa:= front^.data; p:= front;
                   front:= front^.next;  dispose(p)  end;
end;

procedure Duyet;
var p: queuePtr;
begin if front<>NIL then
  begin p:=front;
        while p<>NIL do begin write(p^.data,'  '); p:=p^.next end;
        writeln;
  end else writeln('Queue rong'); readln;
end;

Begin clrscr; front:=NIL; back:=NIL;
repeat write('1. Push  2. Pop  3. Duyet  4. Thoat. Chon: ');
read(chon);
case chon of
1: begin write('Vao n : '); readln(n); ThemVao(n) end;
2: begin n:= LayRa; if n<>0 then writeln('Phan tu lay ra = ',n)
                   else writeln('Queue rong');
   end;
3: Duyet;
end;
until chon=4;
end.
```

7. Cây tìm kiếm nhị phân

Cây nhị phân tổng quát là một tập hợp hữu hạn các đỉnh được xác định đệ quy như sau:

- Một tập trống là một cây nhị phân.
- Giả sử T_1 và T_2 là hai cây nhị phân không cắt nhau và r là một đỉnh mới không thuộc T1, T2. Khi đó ta có thể thành lập một cây nhị phân mới T với gốc r có T1 là cây con bên trái, T2 là cây con bên phải của gốc.

Cây nhị phân có thể trống, mỗi đỉnh của nó luôn luôn có hai cây con là cây con bên trái và cây con bên phải. Mỗi đỉnh của cây nhị phân chỉ có nhiều nhất là hai đỉnh con: một đỉnh con bên trái (đó là gốc của cây con trái) và một đỉnh con bên phải (đó là gốc của cây con phải). Ví dụ cây nhị phân:



Hình 1

Cách cài đặt cây nhị phân:

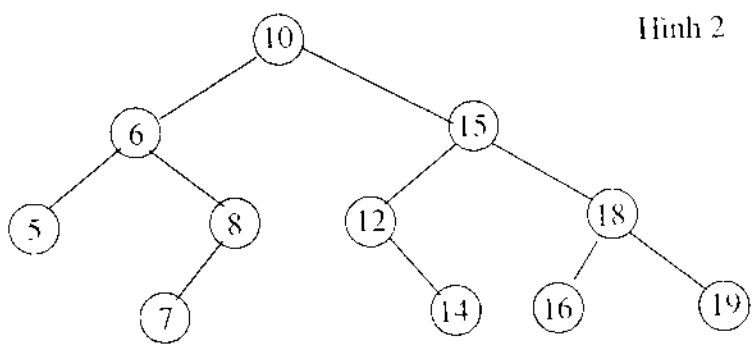
```
type  nodePtr= ^node;
      node= record
              key: integer;
              left,right: NodePtr
            end;
var root: nodePtr;
```

Con trỏ root trỏ tới gốc cây. Con trỏ left trỏ tới đỉnh con bên trái (nếu cây con bên trái là rỗng thì left = NIL), con trỏ right trỏ tới đỉnh con bên phải. Thông tin về mỗi đỉnh: giá trị khoá của đỉnh và các thông tin cần thiết khác.

Trong các thuật toán thực hiện bằng máy tính, khi cần tìm kiếm nhanh trên một tập các đối tượng nào đó, ta nên nghĩ ngay tới việc xác định cho mỗi đối tượng một khoá và sắp xếp chúng trên một cây tìm kiếm nhị phân. **Cây tìm kiếm nhị phân** là cây nhị phân hoặc trống, hoặc thoả mãn 3 điều kiện sau: khoá của các đỉnh thuộc cây con trái nhỏ hơn khoá của gốc, khoá của gốc nhỏ hơn khoá của các đỉnh thuộc cây con phải của gốc, cây con trái và cây con phải của gốc cũng là cây tìm kiếm nhị phân.

Hình 2 biểu diễn một cây tìm kiếm nhị phân, trong đó khoá của đỉnh là một số nguyên (khoá của đỉnh cũng có thể là một ký tự hoặc một xâu ký tự).

Hình 2



Chương trình **CayTknp.c** cho cách cài đặt một cây tìm kiếm nhị phân và các phép toán cơ bản trên cây. Dòng 2-7: nêu cách cài đặt cây tìm kiếm nhị phân. Mỗi một đỉnh của cây gồm có khoá của đỉnh, con trỏ left trỏ tới đỉnh con bên trái, con trỏ right trỏ tới đỉnh con bên phải (nếu cây con bên phải là trống thì right=NIL). Chương trình cho hai cách tạo một cây tìm kiếm nhị phân: nhập cây từ bàn phím và nhập cây từ một tệp.

Nhập cây từ bàn phím. Chọn chức năng "1. Chèn" của menu chính, vào giá trị khoá của đỉnh mới và đưa vào biến x, gọi thủ tục Chen(). Trong thủ tục này ta dùng biến con trỏ p chạy trên các đỉnh của cây bắt đầu từ gốc. Khi đang ở một đỉnh nào đó, nếu x nhỏ hơn khoá của đỉnh thì p chạy xuống đỉnh con trái, nếu x lớn hơn khoá của đỉnh thì p chạy xuống đỉnh con phải. Nếu tại một đỉnh nào đó, p cần phải chạy xuống đỉnh con trái (phải) nhưng đỉnh đó không có cây con trái (phải) thì ta treo đỉnh mới vào bên trái (phải) đỉnh đó. Điều kiện p=NIL sẽ kết thúc vòng lặp. Chọn nhiều lần chức năng "1. Chèn" ta sẽ nhập được toàn bộ cây từ bàn phím. Bất kỳ lúc nào trong khi nhập nếu ta muốn xem cấu trúc của cây hiện tại chỉ cần chọn chức năng "4. InRong".

Nhập cây từ tệp. Tạo một tệp văn bản Cay.dat. Dòng đầu chứa số đỉnh của cây, các dòng tiếp theo chứa các giá trị khoá của các đỉnh theo thứ tự cần chèn vào cây. Với cây ở Hình 2 tệp Cay.dat có dạng sau:

```
11
10  6  5  8  7  15  12  18  16  19  14
```

Sau khi nhập cây từ tệp (dùng chức năng 8. NhapTep) ta có thể dùng chức năng "1. Chèn" để chèn thêm các đỉnh mới vào cây với dữ liệu nhập vào từ bàn phím.

Duyệt cây theo chiều rộng. Để dễ dàng vẽ lại cây trên giấy ta dùng cách duyệt cây theo chiều rộng (chọn chức năng "4. InRong" từ menu chính). Các đỉnh của cây in lần lượt từ mức 1 tới mức cuối cùng, trong một mức in từ bên trái sang phải. Đối với mỗi đỉnh in khoá của đỉnh, kèm theo là các khoá của đỉnh con trái và đỉnh con phải của nó (nếu đỉnh con nào không có thì in NIL). Thứ tự các đỉnh của cây trong Hình 2 khi duyệt theo chiều rộng: 10, 6, 15, 5, 8, 12, 18, 7, 14, 16, 19.

Thuật toán duyệt cây theo chiều rộng trong thủ tục InRong: dùng một hàng đợi **h**, lúc đầu **h** chứa đỉnh gốc. Mỗi lần loại một phần tử ở đầu hàng đợi ta thêm vào cuối hàng đợi tất cả các con của nó, làm như vậy cho đến khi hàng đợi rỗng.

Duyệt cây theo thứ tự Preorder (còn gọi là đi qua cây theo chiều sâu). Danh sách các đỉnh của cây theo thứ tự preorder được xác định như sau:

- Nếu T là một cây gồm 1 đỉnh duy nhất thì danh sách preorder chứa 1 đỉnh duy nhất.

- Nếu T là cây có gốc r và các cây con của gốc là T_1, T_2 thì danh sách preorder các đỉnh của T bắt đầu là r, theo sau là các đỉnh của T_1 theo thứ tự preorder, rồi đến các đỉnh của T_2 theo thứ tự preorder.

Thứ tự các đỉnh của cây trong Hình 2 khi duyệt theo thứ tự preorder: 10, 6, 5, 8, 7, 15, 12, 14, 18, 16, 19 (chọn chức năng "5. PreOrder" từ menu chính). Thuật toán trong chương trình là thuật toán đệ quy.

Duyệt cây theo thứ tự Inorder. Danh sách các đỉnh của cây theo thứ tự inorder được xác định như sau:

- Nếu T là một cây gồm 1 đỉnh duy nhất thì danh sách inorder chứa 1 đỉnh duy nhất.

- Nếu T là cây có gốc r và các cây con của gốc là T_1, T_2 thì danh sách inorder các đỉnh của T bắt đầu là các đỉnh của T_1 theo thứ tự inorder, rồi đến gốc r, các đỉnh của T_2 theo thứ tự inorder.

Thứ tự các đỉnh của cây trong Hình 2 khi duyệt theo thứ tự inorder: 5, 6, 7, 8, 10, 12, 14, 15, 16, 18, 19 (chọn chức năng "6. Inorder" từ menu chính).

Duyệt cây theo thứ tự Postorder. Danh sách các đỉnh của cây theo thứ tự postorder được xác định như sau:

- Nếu T là một cây gồm 1 đỉnh duy nhất thì danh sách postorder chứa 1 đỉnh duy nhất.

- Nếu T là cây có gốc r và các cây con của gốc là T_1, T_2 thì danh sách postorder các đỉnh của T bắt đầu là các đỉnh của T_1 theo thứ tự postorder, các đỉnh của T_2 theo thứ tự postorder, sau cùng là gốc r.

Thứ tự các đỉnh của cây trong Hình 2 khi duyệt theo thứ tự postorder: 5, 7, 8, 6, 14, 12, 16, 19, 18, 15, 10 (chọn chức năng "7. PostOrder").

Tìm kiếm. Tìm trên cây có chứa đỉnh với khoá x cho trước hay không. Thủ tục TimKiem(x, p) dùng một con trỏ p để chạy trên các đỉnh của cây bắt đầu từ gốc. Khi

đang ở một đỉnh nào đó, nếu x trùng với khoá của đỉnh thì tìm kiếm thành công và p là đỉnh cần tìm, nếu x nhỏ hơn khoá của đỉnh thì p chạy xuống đỉnh con trái, nếu x lớn hơn khoá của đỉnh thì p chạy xuống đỉnh con phải. Nếu tìm kiếm không thành công thì p = NIL. Nếu tìm kiếm thành công thì p chứa địa chỉ của đỉnh có khoá là x.

Loại bỏ một đỉnh khỏi cây. Chức năng "2. Xoá" nhằm loại bỏ khỏi cây một đỉnh có khoá x cho trước (khoá x nhập từ bàn phím). Trước tiên chương trình tìm kiếm đỉnh có giá trị khoá bằng x. Thủ tục Xoa(p) làm nhiệm vụ loại bỏ khỏi cây một đỉnh do con trỏ p trỏ tới.

Thuật toán loại bỏ một đỉnh. Nếu đỉnh cần loại bỏ là lá thì ta chỉ cần cắt lá đó đi. Nếu đỉnh cần loại bỏ có một trong hai cây con là cây trống, ta chỉ cần treo cây con khác trống vào vị trí của đỉnh bị loại. Nếu cả hai cây con của đỉnh cần loại bỏ đều khác trống thì ta thay khoá của đỉnh cần loại bỏ bởi khoá của đỉnh ngoài cùng bên phải của cây con trái (chính là đỉnh có khoá lớn nhất của cây con trái), rồi loại bỏ đỉnh ngoài cùng bên phải của cây con trái. Sau khi loại bỏ đỉnh theo cách như vậy thì cây vẫn là cây tìm kiếm nhị phân.

```
uses crt;
type  nodePtr= ^node;
      node= record
              key: integer;
              left,right: NodePtr
            end;
var root,tk: nodePtr; x,chon,i,n: integer;
    a: array[1..100] of integer; f: text;

procedure Chen(x: integer);
var p,q: nodePtr;
begin new(q); q^.key:=x; q^.left:=Nil; q^.right:=nil;
if root=NIL then root:=q else
begin p:=root;
  while p<>NIL do
    begin if x<p^.key then begin if p^.left<>nil then p:=p^.left
                                 else begin p^.left:=q; p:=nil end
                           end
          else begin if x>p^.key then begin
                       if p^.right<>nil then p:=p^.right
                       else begin p^.right:= q; p:=nil end
                                    end
                     else p:=nil;
               end
    end;
end;
end;

procedure TimKiem(x: integer; var p: nodePtr);
var found: boolean;
begin p:= root; found:= false;
while(p<>nil) and (not found) do
  if p^.key=x then found:= true
  else if x<p^.key then p:= p^.left else p:= p^.right;
end;

procedure InRong;
```

```
const max =50;
var truoc,cuoi,dem: integer; p,q: nodePtr;
    h: array[1..max] of nodePtr;
begin h[1]:=root; truoc:=1; cuoi:=1; dem:=1;
while (dem>0) do
  begin q:=h[truoc]; write(q^.key,' : '); p:= q^.left;
        if p<>nil then begin write(p^.key,' - '); inc(cuoi);
                             h[cuoi]:=p; inc(dem);
                       end else write('NIL - ');
        p:= q^.right;
        if p<>nil then begin writeln(p^.key);
                             inc(cuoi); h[cuoi]:=P; inc(dem);
                       end else writeln('NIL');
        h[truoc]:=nil; inc(truoc); dec(dem);
  end;
end;

procedure Xoa(var p: nodePtr);
var q,q1: nodePtr;
begin
if p^.right=NIL then begin q:=p; p:=p^.left end
else if p^.left=NIL then begin q:=p; p:=p^.right end
     else begin q:= p^.left;
               if q^.right=nil then begin p^.key:=q^.key;
                                          p^.left:=q^.left
                                    end
               else begin
                      repeat q1:=q; q:=q^.right until q^.right=nil;
                      p^.key:=q^.key; q1^.right:= q^.left;
                    end
          end;
dispose(q);
end;

procedure PreOrder(p: nodePtr);
begin if p<> nil then
begin writeln(p^.key); PreOrder(p^.left); PreOrder(p^.right) end;
end;

procedure InOrder(p: nodePtr);
begin if p<> nil then
begin InOrder(p^.left);writeln(p^.key); InOrder(p^.right) end;
end;

procedure PostOrder(p: nodePtr);
begin if p<> nil then
begin PostOrder(p^.left); PostOrder(p^.right); writeln(p^.key) end;
end;

begin root:=nil; clrscr;
repeat
writeln('1.Chen    2.Xoa       3.TimKiem  4.InRong  5.PreOder ');
  write('6.InOrder 7.PostOrder 8.NhapTep  9.Exit.   Chon:');
readln(chon);
case chon of
1: begin write('Nhap khoa moi: '); readln(x); Chen(x);
   end;
```

```
2: begin write('Nhap khoa can xoa: '); readln(x);
    TimKiem(x,tk);
    if tk<>nil then Xoa(tk)
    else writeln('Khong tim thay gia tri khoa');
   end;
3: begin write('Nhap khoa can tim : '); readln(x);
    TimKiem(x,tk);
    if tk<>NIL then writeln('Da tim thay dinh co khoa : ',tk^.key)
                    else writeln('Khong tim thay dinh co khoa nay');
   end;
4: InRong;
5: begin writeln('Danh sach PreOrder: '); PreOrder(Root)  end;
6: begin writeln('Danh sach InOrder: '); InOrder(Root) end;
7: begin writeln('Danh sach PostOrder: '); PostOrder(Root) end;
8: begin assign(f,'Cay.dat'); reset(f); readln(f,n);
         for i:=1 to n do begin read(f,a[i]); Chen(a[i]) end;
         close(f); write('Da nhap xong cay tu tep...'); readln;
   end;
end
until chon=9;
end.
```

Ta có nhận xét rằng, thời gian thực hiện phép toán tìm kiếm là số phép so sánh giá trị khoá x cho trước với khoá của các đỉnh nằm trên đường đi từ gốc tới một đỉnh cần tìm, tức là bằng độ dài đường đi từ gốc tới đỉnh cần tìm. Trong trường hợp tốt nhất, mỗi đỉnh của cây đều có hai con trừ mức thấp nhất thì thời gian thực hiện phép toán tìm kiếm là O(logn). Trong trường hợp xấu nhất cây tìm kiếm nhị phân suy biến thành danh sách liên kết (xảy ra khi tạo cây bằng cách chèn vào cây trống lần lượt các đỉnh với các giá trị khoá đã được sắp xếp theo thứ tự tăng dần) thì thời gian thực hiện phép toán tìm kiếm là O(n). Trong trường hợp tổng quát ta xây dựng cây từ cây trống bằng cách chèn vào các đỉnh với các giá trị khoá được sắp xếp một cách ngẫu nhiên thì có thể chứng minh được rằng thời gian trung bình để thực hiện các phép toán trên cây tìm kiếm nhị phân là O(logn).

Bài tập

1. Sử dụng hàm Randomize và hàm Random (hàm tạo một số ngẫu nhiên trên đoạn [0, 1] để tạo hai mảng số ngẫu nhiên A và B, mỗi mảng gồm 10.000 phần tử. In hai mảng này ra tệp văn bản, mỗi dòng 5 số, mỗi số có 6 số lẻ.

2. Khai 4 mảng số thực A, B, C và D. Mỗi mảng có N phần tử, N <= 10.000. Tạo số liệu ngẫu nhiên cho các mảng A và B. Tính mảng C và D theo công thức: C[i] = A[i] + B[i], D[i] = A[i] - B[i] với i = 1, . . ., N. In các mảng C và D ra màn hình hay ra tệp, mỗi dòng 5 số, mỗi số có 5 số lẻ.

3. Lập chương trình nhân hai ma trận A(m,n) và B(n,p) để được ma trận tích C(m,p). Các mảng A, B, C đều khai dưới dạng biến con trỏ và mỗi mảng khai có 10.000 phần tử.

4. Lập chương trình tính nghịch đảo ma trận vuông A có n hàng n cột. Yêu cầu mảng A khai dưới dạng con trỏ, n khai tối đa là 100.

5. Một danh sách được thể hiện bằng mảng có mô tả như sau:

```
const max = 100;
type danh_sach = record
                   a: array[1..max] of integer;
                   last: integer ;
                 end;
var DS : danh_sach;
```

Hãy viết các thủ tục thực hiện các nhiệm vụ:
- Tạo một danh sách rỗng.
- Nhập các phần tử cho danh sách.
- In toàn bộ các phần tử của danh sách.
- Chèn một phần tử mới vào vị trí thứ p của danh sách.
- Xoá một phần tử ở vị trí thứ p của danh sách.
- Xoá các phần tử có giá trị bằng x ra khỏi danh sách.
- Loại bỏ các phần tử lặp ra khỏi danh sách. Ví dụ nếu danh sách là 3, 7, 5, 7, 9, 3, 5, 6 thì sau khi xoá còn: 3, 7, 5, 9, 6.

6. Một danh sách được thể hiện bằng danh sách liên kết có mô tả như sau:

```
type NodePtr = ^Node;
     Node = record
              so : integer;
              next : NodePtr;
            end;
var list : NodePtr ;
```

Hãy viết các thủ tục thực hiện các nhiệm vụ:
- Tạo một danh sách rỗng.
- Nhập các phần tử cho danh sách.
- Chèn một phần tử vào sau phần tử có giá trị x của danh sách.
- Chèn một phần tử vào trước phần tử có giá trị x của danh sách.
- Xoá một phần tử đi sau phần tử có giá trị x của danh sách.
- Xoá các phần tử có giá trị bằng x ra khỏi danh sách.
- Loại bỏ các phần tử lặp ra khỏi danh sách.

7. Một đa thức được biểu diễn bằng mảng có khai báo như sau:

```
const  max = 50 ;
type  so_hang = record
                  he_so : real ;
                  bac : integer;
                end ;
      da_thuc = record
                  Pt: array[1..max] of so_hang;
                  n : integer;
                end;
var P, Q, T : da_thuc;
```

trong đó n là số số hạng của đa thức, mỗi số hạng (gồm có hệ số và bậc) được lưu vào một phần tử của mảng Pt và theo thứ tự bậc giảm dần.

Viết các thủ tục: nhập vào từ bàn phím một đa thức (dùng thủ tục này để nhập hai đa thức P và Q), tính tổng hai đa thức (dùng thủ tục này để cộng hai đa thức P và Q được đa thức T), in một đa thức.

8. Một đa thức được biểu diễn bằng danh sách liên kết có khai báo như sau:

```
type    so_hangPtr = ^so_hang;
        so_hang = record
                    he_so : real ;
                    bac : integer ;
                    next : so_hangPtr;
                  end;
var P : so_hangPtr ;
```

Đa thức P xác định bởi một con trỏ P chứa địa chỉ của số hạng đầu tiên. Hãy lập trình nhập vào hai đa thức (các số hạng xếp theo bậc giảm dần); cho phép lựa chọn một trong bốn phép tính trên hai đa thức: cộng hai đa thức, trừ, nhân và chia; in đa thức kết quả cũng là một danh sách liên kết.

9. Tạo danh sách liên kết thuận có phần dữ liệu của mỗi phần tử là các số kiểu integer. Yêu cầu danh sách luôn luôn được sắp theo chiều tăng của các số, tức là mỗi khi thêm một số mới vào danh sách ta phải tìm đúng vị trí trên danh sách để chèn.

10. Viết hàm nhập một hàng đợi (gồm các số nguyên) dưới dạng danh sách liên kết, dùng hàm này nhập hai hàng đợi Q1 và Q2. Viết hàm trộn hai hàng đợi Q1 và Q2 để được một hàng đợi Q bao gồm các phần tử của Q1 và Q2 nhưng được xếp theo chiều tăng dần. Ví dụ:

Q1 : 50, 90, 100, 200
Q2 : 45, 55, 80, 95, 105
Q : 45, 50, 55, 80, 90, 95, 100, 105, 200

11. Tính giá trị biểu thức. Trong chương trình ta viết một biểu thức toán học x = 89.32 * (345.123 + 12.4) - 100.5 / 25.79, làm thế nào máy tính tính được biểu thức này ? Để đơn giản ta chỉ xét các biểu thức số học chứa các phép toán hai toán hạng : + - * / và hai dấu ngoặc tròn. Khi tính giá trị biểu thức các phép toán trong ngoặc được thực hiện trước, rồi đến các phép toán nhân và chia, sau đó đến cộng và trừ. Trong cùng một mức ưu tiên các phép toán thực hiện từ trái sang phải.

Biểu thức số học Balan là cách viết khác của một biểu thức số học thông thường, trong cách viết thông thường phép toán được đặt giữa hai toán hạng (ví dụ a+ b, a*b) còn trong cách viết Balan phép toán đặt sau các toán hạng (ab+ và ab*). Biểu thức số học Balan không chứa các dấu ngoặc. Cách viết Ba lan của biểu thức ở trên là :

89.32 345.123 12.4 + * 100.5 25.79 / -

Thuật toán xác định giá trị của biểu thức số học Balan. Ta dùng một ngăn xếp S để lưu giữ các toán hạng và các kết quả tính toán trung gian. Đọc lần lượt các thành phần của biểu thức Balan từ trái sang phải, nếu gặp toán hạng thì đẩy nó vào ngăn xếp. Nếu gặp phép toán thì rút hai toán hạng ở đỉnh ngăn xếp ra và thực hiện phép toán này, kết quả nhận được lại đẩy vào ngăn xếp. Lặp lại quá trình cho tới khi toàn bộ biểu thức được đọc qua, lúc đó đỉnh của ngăn xếp chứa giá trị của biểu thức.

Thuật toán chuyển biểu thức số học thông thường E sang biểu thức số học Ba lan E1. Ta dùng một ngăn xếp S để lưu giữ các dấu ngoặc (và các dấu phép toán. Ta đưa vào ký hiệu $ để đánh dấu đáy của ngăn xếp, khi đỉnh của ngăn xếp chứa $ thì ngăn xếp rỗng. Trên tập hợp các ký hiệu $ (+ - * / ta xác định hàm Pri (hàm ưu tiên) như sau: Pri(S) < Pri(( ) < Pri(+) < Pri(-) < Pri(*) = Pri(/). Thuật toán gồm các bước:

Bước 1. Đọc một thành phần của biểu thức E (đọc từ trái sang phải), giả sử thành phần đọc được là x.

1.1. Nếu x là toán hạng thì viết nó vào bên phải biểu thức E1.
1.2. Nếu x là dấu mở ngoặc (thì đẩy nó vào ngăn xếp.
1.3. Nếu x là một trong các phép toán + - * / thì
   a. Xét phần tử y ở đỉnh ngăn xếp.
   b. Nếu Pri(y) >= Pri(x) thì loại y khỏi ngăn xếp, viết y vào bên phải E1 và quay lại điểm a.
   c. Nếu Pri(y) < Pri(x) thì đẩy x vào ngăn xếp.
1.4. Nếu x là dấu đóng ngoặc) thì
   a. Xét phần tử y ở đỉnh ngăn xếp.
   b. Nếu y là dấu phép toán thì loại y khỏi ngăn xếp, viết y vào bên phải E1 và quay lại điểm a.
   c. Nếu y là dấu mở ngoặc (thì loại nó khỏi ngăn xếp.

Bước 2. Lặp lại bước 1 cho tới khi toàn bộ biểu thức E được đọc qua.

Bước 3. Loại phần tử ở đỉnh ngăn xếp và viết nó vào bên phải E1. Lặp lại bước này cho đến khi ngăn xếp rỗng.

Lập chương trình nhập vào từ bàn phím một biểu thức số học thông thường (gồm các số thực hay nguyên, dấu đóng mở ngoặc, dấu cộng, trừ, nhân, chia), tính giá trị biểu thức. Cần lập hai hàm: một hàm chuyển từ biểu thức thông thường sang biểu thức Balan, một hàm tính giá trị biểu thức Balan.

12. Một cây tìm kiếm nhị phân được khai báo như sau:

```
type  nodePtr= ^node;
      node= record
              ma_nhan_vien: integer;
              ho_ten : string[25] ;
              luong : real ;
              left,right: NodePtr
            end;
var root: nodePtr;
```

Viết chương trình làm nhiệm vụ:

- Nhập liên tiếp các nhân viên từ bàn phím và chèn vào cây theo mã nhân viên. Khi nhập vào mã nhân viên không được sắp xếp tăng dần hay giảm dần sẵn, mà phải có tính ngẫu nhiên (để cây khỏi bị lệch).
- Duyệt cây theo thứ tự Preorder.
- In ra màn hình người có lương ít nhất, người có lương nhiều nhất, tổng lương của tất cả mọi người.

Chương 9

Unit Graph

Màn hình máy tính có hai chế độ hiển thị thông tin: chế độ văn bản và chế độ đồ hoạ. Trong chế độ đồ hoạ màn hình chia thành lưới vuông các điểm ảnh (pixel). Màn hình thông dụng hiện nay (VGA) trong chế độ đồ hoạ với độ phân giải cao có 640 cột điểm ảnh đánh số từ 0 đến 639 (ứng với trục X hướng sang phải) và 480 hàng điểm ảnh đánh số từ 0 đến 479 (ứng với trục Y hướng từ trên xuống dưới). Ta có thể vẽ (cho điểm sáng) và tô màu từng điểm ảnh để tạo một hình bất kỳ trên màn hình. Trong chế độ đồ hoạ màn hình cũng có con trỏ vẽ nhưng nó không hiện.

Để làm việc trong chế độ đồ hoạ trên đĩa phải có tệp **Graph.tpu** (chứa các hàm và thủ tục về đồ hoạ thuộc Unit Graph), tệp **EgaVga.bgi** (trình điều khiển màn hình VGA) và 10 tệp **chr** chứa các phông chữ véc tơ (Trip.chr, Litt.chr, Sans.chr, Goth.chr, Scri.chr, Simp.chr, Tscr.chr, Lcom.chr, Euro.chr, Bold.chr). Bắt đầu một chương trình đồ hoạ bao giờ cũng phải có lệnh: **uses graph;**

1. Khởi động chế độ đồ hoạ

• **Thủ tục InitGraph(var gd, gm: integer; path: string)**: khởi động chế độ đồ hoạ (khai báo loại màn hình và độ phân giải). Tham số **path** chứa đường dẫn tới thủ mục lưu tệp BGI, nếu tệp BGI để ở thư mục hiện hành thì path là xâu rỗng. Với màn hình thông dụng VGA hiện nay tham số **gd** (graphic driver, trình điều khiển màn hình) lấy bằng hằng số VGA=9. Với màn hình VGA tham số về độ phân giải màn hình **gm** (graphic mode, mode màn hình) có thể nhận các hằng số: VgaLo= 0 (200x640 điểm ảnh), VgaMed= 1 (350x640 điểm), VgaHi = 2 (480x640 điểm). Các tham số trong lời gọi thủ tục có thể viết ở dạng tên hằng hoặc giá trị số. Nếu trước khi gọi thủ tục này ta gán biến gd bằng hằng số Detect (**gd:=Detect;**) thì máy sẽ tự động tìm chương trình điều khiển thích hợp với loại màn hình đang sử dụng và cho gm nhận giá trị ứng với độ phân giải cao nhất.

• **Hàm GraphResult: integer** cho mã lỗi của lời gọi đồ hoạ cuối cùng. Nếu không có lỗi hàm trả về giá trị 0 (hay hằng GrOK), trái lại cho giá trị khác 0.

• **Hàm GraphErrorMsg(MaLoi): string** cho xâu ký tự giải thích lỗi bằng tiếng Anh của lỗi có mã lỗi là MaLoi. Hàm GraphErrorMsg(GraphResult) cho lời giải thích lỗi của lời gọi đồ hoạ cuối cùng.

Để chuyển từ chế độ văn bản (màn hình có 80*25 ký tự) sang chế độ đồ hoạ (màn hình có 640*480 điểm ảnh với màn hình VGA ở độ phân giải cao) ta dùng mẫu các lệnh:

```
uses Graph ;
var gd,gm: integer ;
begin gd:=Detect; { hoac gd := VGA; gm := VgaHi; }
InitGraph(gd,gm,'');
if GraphResult <> erOK then
    begin writeln(GraphErrorMsg(GraphResult)); readln; halt end;
```

• **Hàm GetMaxX** cho toạ độ trục X lớn nhất, **hàm GetMaxY** cho toạ độ trục Y lớn nhất. Điểm giữa màn hình có toạ độ là (getmaxx div 2, getmaxy div 2).

• **Thủ tục ClearDevice**: xoá màn hình đồ hoạ và đưa con trỏ vẽ về vị trí (0,0).

• **Hàm GetX** cho biết toạ độ cột hiện tại của con trỏ vẽ, **Hàm GetY** cho toạ độ dòng hiện tại của con trỏ.

• **Thủ tục MoveToXY**(x,y: Integer) : di chuyển con trỏ vẽ đến toạ độ (x,y).

• **Thủ tục MoveRel(dx,dy: integer)**: giả sử con trỏ đang ở toạ độ (x,y), thủ tục này sẽ di chuyển con trỏ tới toạ độ (x+dx, y+dy).

• **Thủ tục RestoreCrtMode**: chuyển từ chế độ đồ hoạ sang chế độ văn bản mà không tắt chế độ đồ hoạ.

• **Hàm GetGraphMode : integer** trả về mode đồ hoạ hiện tại được đặt bởi InitGraph hay SetGraphMode. Giá trị của mode là từ 0 đến 5 phụ thuộc vào loại màn hình (ví dụ màn hình VGA thì mode là từ 0 tới 2).

• **Thủ tục SetGraphMode(mode: integer)**: thiết lập mode đồ hoạ và xoá màn hình. Giá trị mode cần phải hợp lệ với loại màn hình. Thủ tục này dùng để đặt lại mode màn hình khác với mode đã đặt bởi InitGraph. Lệnh SetGraphMode(GetGraphMode) dùng để chuyển trở lại chế độ đồ hoạ khi đang ở chế độ văn bản tạo bởi lệnh RestoreCrtMode.

• **Hàm GetMaxMode: integer** cho giá trị lớn nhất của mode màn hình ứng với trình điều khiển màn hình hiện hành.

• **Hàm GetDriverName: string** cho tên của trình điều khiển màn hình hiện hành.

• **Hàm GetModeName(Mode: integer): string** cho tên của mode màn hình có số hiệu là Mode của trình điều khiển màn hình hiện hành.

• **Thủ tục CloseGraph** : đóng chế độ đồ hoạ, trở về chế độ văn bản.

Chương trình **KhoiDong.pas** minh hoạ cách dùng các hàm vừa nêu.

```
uses Graph;
var gd,gm,maloi,x1,y1,modeht,modeln:integer;
    smode,sdriver: string[20];

begin gd:=Detect; InitGraph(gd, gm,''); maloi:=GraphResult;
if MaLoi <> grOk then
  begin Writeln('Ma loi = ',maloi, ' : ', GraphErrorMsg(maloi));
        readln; halt;  end
else
begin modeht := GetGraphMode; modeln:= GetMaxMode;
smode := GetModeName(modeht); sdriver := GetDriverName;
LineTo(100,100); readln; x1:= GetX; y1:= GetY;

RestoreCrtMode;
writeln('Ten trinh dieu khien man hinh = ',sdriver);
writeln('Mode man hinh hien tai = ',modeht,', Ten = ',smode);
writeln('Mode man hinh lon nhat = ',modeln);
writeln('Con tro dang o toa do ',x1,'  ',y1);
write('An Enter ve che do do hoa'); readln;

SetGraphMode(GetGraphMode);
MoveTo(100,200); LineTo(200,100); readln;
MoveRel(50,50); LineTo(GetMaxX, GetMaxY); readln;
```

```
ClearDevice; Line(0,GetMaxy,GetMaxx,0); readln;
CloseGraph;
end;
end.
```

2. Sử dụng màu trong đồ hoạ

Số màu hiển thị trên màn hình tuỳ thuộc vào loại màn hình sử dụng, màn hình CGA: 4 màu, màn hình EGA và VGA: 16 màu. Bảng màu (Palette) điều khiển các màu sắc hiển thị trên màn hình, thay đổi bảng màu sẽ dẫn đến thay đổi màu của từng điểm ảnh đã vẽ. Bảng màu thuộc kiểu dữ liệu bản ghi có tên là PletteType đã được unit Graph định nghĩa sẵn:

```
const MaxColors = 15;
type PaletteType = record
                     size: byte ;
                     colors: array[0..MaxColors] of shortint;
                   end;
```

trong đó size là số lượng màu của bảng màu, colors chứa các giá trị màu của bảng màu. Unit Graph dùng các hằng màu như trong Unit Crt: Black=0, . . ., White=15 (xem Bảng ở mục 1 chương 7). Trong chế độ 16 màu các màu có số hiệu từ 0 đến 15. Bảng màu trong đồ hoạ giống như bảng màu của hoạ sĩ khi vẽ.

• **Hàm GetPaletteSize: integer** trả về kích thước của bảng màu trong mode đồ hoạ hiện tại.

• **Hàm GetMaxColor: word** cho số hiệu màu cao nhất có thể truyền cho thủ tục SetColor.

• **Thủ tục GetPalette(var P: Palettetype)**: cho bảng màu hiện tại và cỡ của nó trong một biến P kiểu Palettetype.

• **Thủ tục SetPalette(ColorNum: word; Color: shortint)**: màu thứ ColorNum trong bảng màu sẽ chứa màu mới là Color. Lệnh SetPalette(0, LightCyan) làm cho màu đầu tiên trong bảng màu thành màu xanh lơ sáng. ColorNum có thể thay đổi từ 0 đến 15 phụ thuộc vào trình điều khiển màn hình và mode màn hình hiện tại. Nếu giá trị truyền cho SetPalette không hợp lệ thì GraphResult trả về giá trị grError và bảng màu không thay đổi.

Chương trình **SetPalet.pas**: khởi động đồ hoạ, in ra các giá trị ngầm định: cỡ của bảng màu (16), các giá trị của mảng P.colors (0, 1, 2, 3, 4, 5, 20, 7, 56, 57, 58, 59, 60, 61, 62, 63), màu nền (0), màu chữ (15), cỡ của bảng màu (16), số hiệu màu lớn nhất (15). Cùng màn hình VGA nhưng nếu đặt gm=VgaLo thì các giá trị của mảng P.Colors sẽ khác. Tiếp theo máy in 16 đường kẻ ngang trên màn hình. Do lệnh SePalette sẽ đặt lại các phần tử của bảng màu theo một màu ngẫu nhiên, nên màu của 16 đường kẻ đã vẽ và màu nền thay đổi liên tục một cách ngẫu nhiên.

```
uses crt,graph;
var gd, gm, comau, maumax,h,y: integer; color,mn,mc: word;
    P: PaletteType; s1, s2: string;

begin  gd:= VGA; gm:= VGAHi; InitGraph(gd, gm, ' ');
if GraphResult <> grOk then  Halt(1);
GetPalette(P); mn:=GetBkColor;  mc:= GetColor;
comau:= GetPaletteSize; maumax:= GetMaxColor;
```

```
h:=textheight('H'); y:=10; str(P.size,s1);
outtextxy(50,y,'BANG MAU NGAM DINH,  P.Size = '+s1);
for color:=0 to P.size-1 do
  begin y:= y+h+5; str(color,s1); str(P.colors[color],s2);
        outtextxy(50,y,'P.colors[' + s1+ '] = ' + s2);
  end;

str(mn,s1); y:=y+h+5; outtextxy(50,y,'Mau nen ngam dinh = '+s1);
str(mc,s1); y:=y+h+5; outtextxy(50,y,'Mau chu ngam dinh = '+s1);
str(comau,s1); y:=y+h+5;
outtextxy(50,y,'Co bang mau ngam dinh = '+s1);
str(maumax,s1); y:=y+h+5;
outtextxy(50,y,'Gia tri mau lon nhat ngam dinh = '+s1); readln;

cleardevice; SetLineStyle(0,0,3);
if P.Size <> 1 then
  begin for color := 0 to P.Size-1 do
          begin SetColor(color); Line(0,color*20,639,color*20);
                delay(500)
          end;
    randomize;
    repeat SetPalette(random(P.Size),random(P.Size)); delay(100);
    until KeyPressed;
  end else  Line(0, 0, 600, 0);
readln; CloseGraph;
end.
```

• **Thủ tục SetAllPalette(P: PaletteType)**: định nghĩa lại toàn bộ bảng màu theo các trị được xác định trong biến P có kiểu là PaletteType.

Chương trình **SetAllP.pas** đặt lại toàn bộ bảng màu.

```
uses graph;
var gd, gm: integer; P: PaletteType;
begin  gd:=detect; InitGraph(gd,gm,' ');
if GraphResult <> grOk then Halt(1);
setcolor(blue); SetLineStyle(0,0,3);
Line(0,0,GetMaxX,GetMaxY); readln; {đường blue, nền black ngầm định}
setPalette(1,5); readln; {đường magenta, nền black}
with P do begin Size:=4; Colors[0]:=5; Colors[1]:=3;
                Colors[2]:=1; Colors[3]:=2;
                SetAllPalette(P); {đường cyan, nền magenta}
          end; readln;
cleardevice; readln; {nền magenta, trống}
setcolor(2); setBkColor(3);   {đường blue, nền green}
Line(0,getmaxy,getmaxx,0); readln; closegraph;
end.
```

• **Thủ tục SetBkColor(N: word)**: lấy màu thứ N trong bảng màu làm màu nền mới , N có thể nhận giá trị từ 0 đến 15. Trong bảng màu, màu thứ 0 quy định là màu nền, như vậy lời gọi SetBkColor(Red) tương đương với lời gọi SetPalette(0,Red). Sau khi khởi động đồ hoạ màu nền ngầm định là Black = 0. Màu nền có hiệu lực cho đến khi gặp lệnh SetBkColor mới.

• **Hàm GetBkColor: word** trả về màu nền hiện tại, màu nền thay đổi từ 0 đến 15 phụ thuộc vào trình điều khiển màn hình và mode màn hình hiện tại.

Chương trình **MauNen.pas** minh họa cách dùng Setbkcolor và Getbkcolor:

```
uses Crt, Graph;
var Gd, Gm: Integer; Color: Word; Pal: PaletteType;
begin Gd := Detect; InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then  Halt(1);
Randomize; GetPalette(Pal);
if Pal.Size <> 1 then begin
   repeat  Color := Succ(GetBkColor);
     if Color > Pal.Size-1 then  Color := 0; SetBkColor(Color);
     LineTo(Random(GetMaxX), Random(GetMaxY)); delay(1000);
   until KeyPressed;
                        end
else  Line(0, 0, GetMaxX, GetMaxY);
Readln; CloseGraph;
end.
```

• **Thủ tục SetColor(N : word)**: lấy màu thứ N trong bảng màu làm màu vẽ mới. N có thể nhận giá trị từ 0 tới 15. Màu vẽ ngầm định là White. Màu vẽ có hiệu lực cho đến khi gặp lệnh SetColor mới. Chú ý lệnh SetColor không làm thay đổi bảng màu, còn lệnh SetBkColor làm thay đổi bảng màu.

• **Hàm GetColor: word** trả về màu vẽ hiện tại, màu vẽ có giá trị từ 0 đến 15 phụ thuộc vào trình điều khiển màn hình và mode đồ hoạ hiện tại.

Chương trình **MauVe.pas** minh hoạ cách dùng SetColor và GetColor:

```
uses Graph, Crt;
var Gd, Gm: Integer; Color: Word; Pal: PaletteType;
begin Gd := Detect; InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
Randomize; GetPalette(Pal);
repeat  Color := Succ(GetColor);
if Color > Pal.Size - 1 then  Color := 0;
SetColor(Color); LineTo(Random(GetMaxX), Random(GetMaxY)); delay(50);
until KeyPressed;
CloseGraph;
end.
```

3. Vẽ điểm và đường

• **Thủ tục PutPixel(x,y: integer; color: word)**: vẽ một điểm ảnh theo màu đã xác định bởi color tại toạ độ (x,y).

• **Hàm GetPixel(x,y: integer): word** lấy màu của điểm ảnh tại toạ độ (x,y).

Chương trình **Pixel.pas**: tô ngẫu nhiên các điểm trên màn hình.

```
uses crt, graph;
var gd,gm,x,y,mau: integer;
begin gd:= vga; gm:= vgahi; InitGraph(gd,gm,'');
if GraphResult <> grOk then Halt(1); Randomize;
repeat  x:= Random(640); y:=Random(480) ; mau:=random(16);
if (GetPixel(x,y) <> Black) then PutPixel(x,y,Black)
                            else PutPixel(x,y ,mau);
delay(1);
until KeyPressed;
```

```
readln; CloseGraph;
end.
```

• **Thủ tục Line(x1,y1,x2,y2: integer)**: vẽ đoạn thẳng nối (x1,y1) với (x2,y2), con trỏ không thay đổi vị trí.

• **Thủ tục LineTo(x,y: integer)**: vẽ một đoạn thẳng từ vị trí hiện thời của con trỏ tới điểm có toạ độ (x,y), con trỏ về vị trí mới. Ví dụ về dùng lệnh Line và Lineto có thể xem chương trình KhoiDong.pas của Mục 1, SetPalet.pas và MauNen.pas của Mục 2.

• **Thủ tục LineRel(dx,dy: integer)**: vẽ đoạn thẳng từ vị trí hiện tại (x,y) của con trỏ tới vị trí (x+dx, y+dy) và di chuyển con trỏ tới vị trí mới

• **Thủ tục Rectangle(x1,y1,x2,y2: integer)**: vẽ một đường chữ nhật có các cạnh song song với các cạnh của màn hình, với toạ độ góc trên trái là (x1,y1) và góc dưới phải là (x2,y2). Điều kiện 0<=x1<=x2<= GetMaxx, 0<=y1<=y2<= GetMaxy.

• **Thủ tục SetLineStyle(LineStyle, Pattern, Thickness : word)** thiết lập kiểu vẽ các đường thẳng. Tham số LineStyle dùng để thiết lập kiểu đường và có thể nhận các giá trị: SolidLn = 0 (đường nét liền), DottedLn=1 (đường nét chấm chấm), CenterLn=2 (đường nét chấp gạch), DasheLn=3 (đường nét gạch), UserBitLn=4 (đường do người dùng tự định nghĩa). Tham số Thickness xác định độ lớn của đường, các giá trị có thể nhận: NormWidth=1 (nét bình thường, ngầm định), ThickWidth=3 (nét to).

Tham số Pattern xác định mẫu vẽ đường. Nếu LineStyle = 0, 1, 2 hay 3 thì cho Pattern=0 (ngầm định). Nếu dùng LineStyle=4 thì Pattern là một số gồm 16 bít chỉ kiểu mẫu của đường. Chẳng hạn, nếu mẫu 16 bít là 0001 0001 0001 0001 (hệ 2) = $1111 (hệ 16) thì Pattern = $1111, đường thẳng sẽ là đường chấm chấm thưa (vị trí bit bằng 1 sẽ có chấm điểm)

Chương trình **SetLine.pas** minh hoạ cách dùng thủ tục SetLineStyle:

```
uses graph;
var gd, gm: Integer; x1, y1, x2, y2: integer;
begin  gd := detect;  InitGraph(gd,gm, ' ');
if GraphResult <> grOk then  Halt(1);
x1 := 10; y1 := 10; x2 := 600; y2 := 400;
SetLineStyle(DottedLn, 0, NormWidth);
Rectangle(x1,y1,x2,y2); readln; cleardevice;
SetLineStyle(UserBitLn,$C3,ThickWidth);{0000 0000 1100 0011}
Rectangle(Pred(x1),Pred(y1),Succ(x2),Succ(y2));
readln; CloseGraph;
end.
```

• **Thủ tục GetLineSettings(var LineInfo: LineSettingsType)**: trả về kiểu đường hiện tại, mẫu đường, độ dày của đường mà đã thiết lập bởi SetLineStyle. Kiểu bản ghi LineSettingsType được định nghĩa trong unit Graph như sau:

LineSettingsType = record LineStyle, Pattern, Thickness: word end;

Chương trình **GetLine.pas** minh hoạ cách dùng thủ tục GetLineSettings:

```
uses graph;
var gd, gm: integer; OldStyle: LineSettingsType;
begin Gd := Detect; InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
```

```
Line(0, 100, 639, 100); GetLineSettings(OldStyle); readln;
SetLineStyle(DottedLn,0,ThickWidth);
Line(0, 200, 639, 200);  readln;
with OldStyle do SetLineStyle(LineStyle, Pattern, Thickness);
Line(0, 300, 639, 300); readln; CloseGraph;
end.
```

• **Thủ tục DrawPoly**(N, A) vẽ một đường gấp khúc đi qua N điểm theo kiểu đường và màu vẽ hiện tại. A là mảng gồm N phần tử chứa toạ độ các điểm mút, mỗi phần tử có kiểu PointType được định nghĩa sẵn bởi unit Graph như sau:

PointType = Record X, Y : Integer End;

trong đó X là toạ độ cột của điểm, Y là toạ độ hàng của điểm. Khi điểm cuối trùng với điểm đầu thì ta được một đường gấp khúc khép kín. Trong chương trình mảng các bản ghi A có thể nhập vào từ bàn phím, từ tệp văn bản (nếu nhiều điểm) hoặc gán trực tiếp.

Chương trình **TamGiac.pas**: vẽ một tam giác.

```
uses graph;
const  Triangle: array[1..4] of PointType =
   ((x:100;y:100),(x:600;y:150),(x:320;y:450),(x:100;y:100));
var gd, gm: integer;
begin  gd:=Detect; InitGraph(gd,gm,'');
if GraphResult <> grOk then Halt(1); SetLineStyle(0,0,3);
DrawPoly(SizeOf(Triangle) div SizeOf(PointType), Triangle);
readln; closegraph;
end.
```

• **Thủ tục Circle(x, y: integer; r: word)**: vẽ đường tròn tâm (x,y) và bán kính r.

• **Thủ tục Arc(x, y: integer; gocdau, goccuoi, r: word)** vẽ cung tròn có tâm là *(x,y)*, bán kính *r* từ góc xuất phát là *gocdau* đến góc kết thúc là *goccuoi* , các góc có giá trị từ 0 đến 360 độ.

• **Thủ tục Ellipse(x, y : integer ; gocdau, goccuoi, rx, ry : word)** vẽ một cung ellip có tâm *(x, y)* với bán kính ngang là *rx*, bán kính dọc *ry* từ góc xuất phát *gocdau* đến góc kết thúc *goccuoi*.

Chương trình **Circle.pas** vẽ các đường tròn đồng tâm và các cung tròn nét to.

```
uses graph;
var gd, gm: integer; Radius: integer;
begin gd := Detect; InitGraph(gd, gm, ' ');
if GraphResult <> grOk then  Halt(1);  setlinestyle(0,0,3);
for Radius:=1 to 23 do Circle(320,240,Radius*10);
readln; ClearDevice; SetColor(LightMagenta);
for Radius:=1 to 23 do Arc(320,240,0,280,Radius*10);
readln; closegraph;
end.
```

4. Vẽ và tô miền

• **Thủ tục SetFillStyle(pattern: word ; color: word)**: chọn mẫu tô (pattern) và màu tô (color) cho các hình đặc (các hình vẽ bởi FillPoly, Bar, Bar3D, PieSlice ....). Pattern có giá trị từ 0 đến 12: hằng số EmptyFill hay bằng 0 là tô bằng màu nền, SolidFill = 1 là tô bằng đường nét liền, SlashFill = 4 là tô bằng các đường ///// đậm, CloseDotFill = 11 là tô bằng các dấu chấm sát nhau... Nếu Pattern có giá trị bằng

UserFill=12 thì mẫu và màu tô của người dùng đặt bởi lệnh SetFillPattern có tác dụng. Ngầm định Pattern = 1, Color = White.

• **Thủ tục SetFillPattern(Pattern: FillPatternType; Color: word):** lựa chọn mẫu tô theo người dùng. FillPatternType là kiểu dữ liệu mảng 8 byte được định nghĩa như sau:

```
Type  FillPatternType = array[1..8] of byte;
```

• **Thủ tục GetFillSettings(var FillInfo: FillSettingsType)**: trả về mẫu tô và màu tô hiện tại được thiết lập bởi SetFillStyle hay SetFillPattern. Kiểu dữ liệu FillSettingsType được định nghĩa như sau:

```
Type FillSettingsType = record  Pattern, Color: word; end;
```

• **Thủ tục GetFillPattern(var FillPattern: FillPatternType)**: trả về mẫu tô đã chọn hiện tại được thiết lập bởi SetFillStyle hay SetFillPattern. Nếu người dùng không dùng SetFillPattern thì thủ tục trả về một mảng toàn các số $FF.

Chương trình **KieuTo.pas** trình bày 11 mẫu tô (pattern = 1, 2, . . ., 11)

```
uses graph;
var gd, gm,i: integer; FillInfo: FillSettingsType;
begin gd := Detect; InitGraph(gd,gm,' '); randomize;
if GraphResult <> grOk then Halt(1);
GetFillSettings(FillInfo); Bar(0, 0, 190, 190); readln;
for i:=1 to 11 do begin
  SetFillStyle(i, random(15)+1); Bar(200, 0, 639, 479);  readln end;
with FillInfo do SetFillStyle(Pattern, Color);
Bar(0, 200, 190, 390); readln; CloseGraph;
end.
```

Chương trình **LamKieu.pas**: cất giữ kiểu tô ngầm định, đặt một kiểu tô tự làm (xám 50 %), vẽ và tô màu hình chữ nhật, hoàn lại kiểu tô cũ, in giá trị 8 byte của mẫu tô xám 50% (170, 85, 170, 85, 170, 85, 170, 85 hệ 10).

```
uses Graph;
const  Gray50:FillPatternType=($AA,$55,$AA,$55,$AA,$55,$AA,$55);
var Gd, Gm,i : Integer; OldPattern,MauTo : FillPatternType;
begin  Gd := Detect; InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then Halt(1);
GetFillPattern(OldPattern);  SetFillPattern(Gray50, white);
{ SetFillStyle(12,red); mau to va mau to theo SetFillPattern }
Bar(0, 0, 400, 400); ReadLn; GetFillPattern(MauTo);
SetFillPattern(OldPattern,White); Bar(0, 0, 400, 400); ReadLn;
CloseGraph;
for i:=1 to 8 do writeln('MauTo[',i,'] = ',MauTo[i]); readln;
end.
```

• **Thủ tục FloodFill(x,y: integer; border: word)**: tô màu một miền kín giới hạn bởi một đường bao quanh, trong đó (x,y) là toạ độ một điểm bất kỳ thuộc phần trong của miền cần tô. Biến Border là màu của đường viền bao quanh miền. Mẫu tô và màu tô xác định bởi lệnh SetFillStyle(). Nếu toạ độ (x,y) ở ngoài miền kín thì vùng ngoài miền kín được tô màu. Nếu trên màn hình không có miền kín mà đường viền bao quanh có màu Border hay miền định tô bị hở thì toàn màn hình được tô màu.

Chương trình **FloodF.pas**: vẽ đường chữ nhật và tô màu bên trong, tô màu miền

ngoài một đường tròn.

```
uses Graph;
var gd, gm: integer;
begin gd:= Detect; InitGraph(gd,gm,'');
if GraphResult <> grOk then   Halt(1);
SetColor(Cyan); SetFillStyle(1,Red);
Rectangle(220,140,420,340); readln;
FloodFill(300,220,Cyan); Readln;
Cleardevice; SetColor(white); SetFillStyle(1,Lightblue);
Circle(320,240,200); readln;
FloodFill(30,20,white); readln;
CloseGraph;
end.
```

• **Thủ tục Bar(x1,y1,x2,y2: integer)**: vẽ và tô màu hình chữ nhật với mẫu tô và màu tô xác định bởi lệnh SetFillStyle.

• **Thủ tục Bar3D(x1,y1,x2,y2: integer; dept: word; top: boolean)**: vẽ một khối hộp chữ nhật, mặt trước là hình chữ nhật được tô màu xác định bởi (x1,y1) và (x2,y2) như trong thủ tục Bar. Tham số Dept: độ sâu của khối hộp. Top bằng TopOn: hộp có nắp, Top=TopOff là hộp không nắp.

Chương trình **KhoiHop.pas** : vẽ hình chữ nhật, khối hộp có tô màu các mặt.

```
uses graph;
var gd, gm: integer;
begin  gd:=Detect; InitGraph(gd,gm,'');
if GraphResult <> grOk then Halt(1);
Bar(1,1,100,100);
SetColor(Cyan); SetFillStyle(1,lightred);
Bar3D(150,130,440,470,150,topon);
setFillStyle(8,Blue); FloodFill(300,125,Cyan);
setFillStyle(11,LightGreen); FloodFill(450,165,Cyan);
readln; closegraph;
end.
```

• **Thủ tục FillPoly(N, A)** vẽ và tô màu một đa giác có N đỉnh, mảng A chứa toạ độ các đỉnh như trong thủ tục DrawPoly.

Chương trình **CoSao.pas** vẽ lá cờ Việt nam.

```
uses graph;
const  Sao: array[1..10] of PointType =
   ((x:439;y:201),(x:352;y:196),(x:320;y:115),(x:288;y:196),
    (x:201;y:201),(x:268;y:257),(x:247;y:341),(x:320;y:295),
    (x:393;y:341),(x:372;y:257));
var gd, gm: integer;
begin  gd:=Detect; InitGraph(gd,gm,'');
SetBkColor(LightRed); SetColor(Yellow); SetFillStyle(1,Yellow);
FillPoly(10,Sao); readln; closegraph;
end.
```

• **Thủ tục PieSlice(x, y : integer; gocdau, goccuoi, r : word)** vẽ và tô màu một hình quạt có tâm là *(x,y)* với bán kính *r* từ góc xuất phát *gocdau* đến góc kết thúc *goccuoi*. Mẫu tô và màu tô xác định theo lệnh SetFillStyle.

• **Thủ tục FillEllipse(x, y : integer ; rx , ry : word)** vẽ và tô màu một ellipse có tâm *(x, y)* và các bán kính trục là *rx, ry*. Mẫu tô và màu tô xác định theo thủ tục SetFillStyle.

• **Thủ tục Sector(x, y: integer; gocdau , goccuoi, rx, ry : word)** vẽ và tô màu một phần ellipse có tâm *(x, y)* với các bán kính trục *rx, ry* từ góc xuất phát *gocdau* đến góc kết thúc *goccuoi*. Mẫu tô và màu tô xác định theo lệnh SetFillStyle.

Chương trình **MienTron.pas** minh hoạ cách dùng 3 thủ tục trên.

```
uses Graph;
var gd, gm: integer;
begin gd:= Detect; InitGraph(gd,gm,'');
SetColor(Cyan); SetFillStyle(1,Red);
FillEllipse(320,240,250,150); readln; Cleardevice;
SetColor(LightGreen); SetFillStyle(1,LightGreen);
FillEllipse(320,240,200,200); readln; Cleardevice;
SetColor(White); SetFillStyle(1,LightMagenta);
PieSlice(320,240,15, 360-15,200); readln; Cleardevice;
SetColor(LightRed); SetFillStyle(1,LightBlue);
Sector(320,240,30,360-30,250,200); readln; CloseGraph;
end.
```

• **Thủ tục GetAspectRatio(var Xasp, Yasp: word)**: cho hai số nguyên để xác định tỷ lệ co dãn giữa chiều ngang và chiều dọc của màn hình Xasp / Yasp trong chế độ đồ hoạ hiện tại. Ví dụ với màn hình VGA (gd=9) thì hai số này phụ thuộc vào cách đặt gm (độ phân giải màn hình): Gm=0 thì Xasp=4500 và Yasp=10000, Gm=1 thì Xasp=7750 và Yasp=10000, Gm=2 thì Xasp=10000 và Yasp=10000. Chương trình **Ratio.pas** minh hoạ thủ tục này:

```
uses Graph;
var Gd, Gm: Integer; Xasp, Yasp: Word; S1,S2: string[10];
    XSideLength, YSideLength: Integer;
begin Gd:=9; Gm:=0; InitGraph(Gd,Gm,'');
if GraphResult <> grOk then   Halt(1);
GetAspectRatio(Xasp, Yasp); str(Xasp:6,s1); str(Yasp:6,s2);
outTextXY(10,10,'Xasp = '+s1+'  Yasp= '+s2);
XSideLength := 300; YSideLength:=300;
YSideLength := Round((Xasp/Yasp)*XSideLength);                    {1}
Rectangle(0, 0, XSideLength, YSideLength); Readln; CloseGraph;
end.
```

Nếu không có lệnh (1) thì chiều đứng của hình vuông sẽ dài hơn nhiều so với chiều ngang (mặc dù ta đều gán hai chiều là 300 điểm). Nhờ hai hằng số Xasp, Yasp ta chỉnh lại chiều dọc để được hình vuông (xem lệnh (1)).

5. Cửa sổ trong chế độ đồ hoạ

• **Thủ tục SetViewPort(x1,y1,x2,y2: Integer; Clip: Boolean)**: định nghĩa một ViewPort (một vùng chữ nhật trên màn hình) giới hạn bởi góc trên trái là (x1,y1) và góc dưới phải là (x2,y2). Nếu Clip là hằng ClipON thì cấm vẽ ở ngoài ViewPort, nếu Clip bằng ClipOff thì có thể vẽ bên ngoài ViewPort. Sau khi gọi thủ tục này toạ độ (0,0) của tất cả các thủ tục vẽ là góc trên trái của Viewport chứ không phải là góc trên bên trái màn hình. Do đó nhờ viewport và cho phép vẽ ra ngoài ta có thể tạo được toạ

độ âm dương trên màn hình.

• **Thủ tục GetViewSettings(var Vp: ViewPortType)**: lấy các thông số của viewport hiện hành đưa vào biến VP, kiểu ViewPortType được định nghĩa như sau:

```
Type ViewPortType= Record x1,y1,x2,y2: word; Clip: boolean; end;
```

• **Thủ tục ClearViewPort**: xoá vùng màn hình giới hạn bởi ViewPort.

• **Thủ tục GraphDefaults** thường được gọi ngay sau thủ tục ClearViewPort để khởi tạo lại tất cả các giá trị mặc định của ViewPort ngầm định (toàn màn hình).

Chương trình **ViewPort.pas**: tạo một hệ toạ độ âm dương, vẽ tam giác vuông ở góc dưới bên phải màn hình, vẽ tam giác vuông đối xứng qua gốc toạ độ (điểm giữa màn hình), vẽ tam giác vuông đối xứng qua trục Y, vẽ tam giác vuông đối xứng qua trục X. Biến VP1 và VP3 chứa thông tin Viewport ngầm định, VP2: thông tin viewport tự đặt.

```
uses Graph;
var gd, gm,x1,y1,x2,y2,x3,y3: integer;  VP1,VP2,VP3: ViewPortType;
begin gd := VGA; gm:=VgaHi; InitGraph(gd,gm,'');
if GraphResult <> grOk then Halt(1); GetViewSettings(VP1);
SetViewPort(319,239,639,479,ClipOff); GetViewSettings(VP2);
Line(-319,0,319,0); Line(0,-239,0,239);
x1:=30; y1:=30; x2:=300; y2:=30; x3:=30; y3:=210;
Line(x1,y1,x2,y2); Line(x1,y1,x3,y3); Line(x2,y2,x3,y3); readln;
Line(-x1,-y1,-x2,-y2);Line(-x1,-y1,-x3,-y3);
Line(-x2,-y2,-x3,-y3); readln;
Line(-x1,y1,-x2,y2);Line(-x1,y1,-x3,y3);Line(-x2,y2,-x3,y3); readln;
Line(x1,-y1,x2,-y2);Line(x1,-y1,x3,-y3);Line(x2,-y2,x3,-y3); readln;
ClearViewPort; readln;
GraphDefaults; GetViewSettings(VP3); readln; ClearDevice;
OutTextXY(130,240,'An ENTER de ket thuc chuong trinh'); Readln;
CloseGraph; with VP1 do writeln(x1,' ',y1,' ',x2,' ',y2,' ',Clip);
with VP2 do writeln(x1,'  ',y1,'  ',x2,'  ',y2,'  ',Clip);
with VP3 do writeln(x1,'  ',y1,'  ',x2,'  ',y2,'  ',Clip);
readln; end.
```

6. Viết chữ trong chế độ đồ hoạ

Trong chế độ đồ hoạ ta có thể viết lên màn hình các dòng chữ với các phông chữ và kích cỡ khác nhau, dòng chữ cũng có thể viết theo chiều dọc màn hình.

• **Thủ tục SetTextStyle(Font, Direction, Size: word)** đặt kiểu phông chữ, hướng viết, cỡ chữ. Tham số **Font** nhận các giá trị: DefaultFont = 0 (phông chữ ngầm định của hệ thống, không cần tệp chr), TriplexFont = 1 (cần tệp Trip.chr), SmallFont = 2 (Litt.chr), SansSerifFont = 3 (Sans.chr), GothicFont = 4 (Goth.chr), ScriptFont = 5 (Scri.chr), SimplexFont = 6 (Simp.chr), TriplexScriptFont = 7 (Tscr.chr), ComplexFont = 8 (Lcom.chr), EuropeanFont = 9 (Euro.chr), BoldFont = 10 (Bold.chr). Tham số **Direction** nhận các giá trị: HorizDir = 0 (viết theo hàng ngang, ngầm định), VertDir = 1 (viết dọc màn hình). **Size** nhận giá trị từ 1 đến 10 (độ phóng của phông chữ). Với font hệ thống cỡ chữ bình thường là 1.

Chương trình **Font.pas** in ra 11 loại phông chữ với tất cả các cỡ từ 1 đến 10. Mười tệp phông chữ CHR để ở thư mục C:\TP\BGI.

```
uses Graph;
var gd, gm, font,y, size: integer; s1,s2:string[4];
begin gd := detect; InitGraph(gd, gm, 'c:\tp\bgi');
 if GraphResult <> grOk then   Halt(1); SetColor(Cyan);
for font:=0 to 10 do
begin  ClearDevice; y := 0;
for size := 1 to 10 do
 begin str(font,s1);str(size,s2);SetTextStyle(font,HorizDir,Size);
   OutTextXY(0,Y,'Font='+s1+' size='+s2); inc(y,TextHeight('H')+1);
 end; Readln;
end; CloseGraph;
end.
```

• **Thủ tục OutTextXY(x, y: integer; St : string)** hiện nội dung của xâu ký tự St theo vị trí (x,y).

• **Thủ tục OutText(St: string)** hiện xâu St theo vị trí hiện hành của con trỏ.

• **Thủ tục SetTextJustify(Horiz, Vert: word)** quy định nơi hiển thị xâu ký tự của OutText theo quan hệ với vị trí hiện tại của con trỏ (giả sử là điểm M=(x,y)), hay của OutTextXy theo quan hệ với toạ độ đã chỉ định M=(x,y). Tham số **Horiz** nhận các giá trị: LeftText=0 (điểm M nằm phía trái xâu, ngầm định). CenterText=1 (điểm M nằm giữa xâu theo chiều ngang). RightText=2 (điểm M nằm phía phải xâu). Tham số **Vert** nhận giá trị : BottomText=0 (điểm M nằm dưới xâu ký tự). CenterText=1 (điểm M nằm giữa xâu theo chiều dọc). TopText=2 (điểm M nằm phía trên xâu, ngầm định).

Chương trình **Justify.pas** biểu diễn tất cả các cách căn chỉnh xâu ký tự theo quan hệ với điểm (x,y). Trong chế độ đồ hoạ ta vẫn có thể dùng các lệnh Readln để nhập dữ liệu, Write và Writeln để in dữ liệu ra màn hình, để làm được điều này ở đầu chương trình phải có lệnh **uses crt** và biến **DirectVideo** của unit crt phải đặt lại bằng False (để bỏ quyền trực tiếp truy nhập của unit Crt vào vùng Video Ram). Chương trình in tất cả các thông số ngầm định về phông chữ sau khi khởi động đồ hoạ, vào chiều dài và chiều rộng hình chữ nhật từ bàn phím và in ra diện tích của nó.

```
uses crt,graph;
var gd, gm,x,y,n,d: integer; s1,s2: string[4];
    ts: TextSettingsType;

begin gd:=Vga; gm:= VgaHi; InitGraph(Gd,Gm,'c:\tp\bgi');
if GraphResult <> grOk then Halt(1);

GetTextSettings(ts); DirectVideo:= False;
write('Font = ',ts.font); writeln('  Direction = ',ts.direction);
writeln('CharSize = ',ts.charsize);
writeln('Horiz = ',ts.horiz,'  Vert = ',ts.vert);
write('Vao chieu dai va chieu rong : '); readln(n,d);
write('Dien tich hinh chu nhat la : ',n*d); readln;

ClearDevice; SetTextStyle(0,0,5);
for n:=0 to 2 do for d:=0 to 2 do
  begin SetTextJustify(n,d); Str(n,s1); Str(d,s2);
        x:= 320 ; y:= 240;  SetColor(Cyan);
        OutTextXY(x,y,'N='+s1+' D='+s2);  FillEllipse(x,y,3,3);
        Readln; ClearDevice;
  end; CloseGraph;
end.
```

• **Hàm TextHeight(St: string)** trả về độ cao của xâu St tính bằng điểm.

• **Hàm TextWidth(St: string)** trả về chiều rộng của xâu St tính bằng điểm.

Chương trình **TextH.pas** : in xâu 'VIET NAM' theo các cỡ lớn dần trên các dòng cho đến khi xâu rộng quá màn hình thì kết thúc.

```
uses graph;
var gd, gm: Integer;
 y: integer; s: string; size: integer;
begin gd := detect; InitGraph(gd, gm,'');
if GraphResult <> grOk then   Halt(1);
y := 0; s := 'VIET NAM'; size := 1;
while TextWidth(s) < GetMaxX do
 begin  OutTextXY(0, y, s);
   inc(y, TextHeight('H')+3); inc(size);
   SetTextStyle(DefaultFont, HorizDir, size);
 end;
readln; CloseGraph;
end.
```

• **Thủ tục SetUserCharSize(MultX, DivX, MultY, DivY: word)**: thay đổi chiều rộng và chiều cao của phông chữ đang dùng. Sau lệnh này chiều rộng của phông chữ hiện hành được nhân với (MultX / DivX), chiều cao được nhân với (MultY / DivY). Thủ tục này chỉ áp dụng được với các phông véc tơ (tham số Font trong thủ tục SetTextStyle bằng 1, 2, . . ., 10).

Chương trình **UserChar.pas** : phông hiện tại là BoldFont cỡ 5, lệnh (1) phóng to phông chữ này theo chiều rộng lên 3 lần, theo chiều cao lên 5 lần.

```
uses Graph;
var gd,gm,y: Integer;
begin gd:= Detect; InitGraph(gd,gm,'d:\tp\bgi');
if GraphResult <> grOk then   Halt(1);
SetTextStyle(10, HorizDir, 5); y:=10;
OutTextXY(10,y,'NORMAL - normal  '); readln;
y:=y+ TextHeight('H')+2; SetUserCharSize(1, 3, 1, 1);
OutTextXY(10,y,'SHORT - short  '); readln;
y:=y+ TextHeight('H')+2; SetUserCharSize(3, 1, 5, 1);          {1}
OutTextXY(10,y,'WIDE-w'); Readln; CloseGraph;
end.
```

7. Tạo ảnh chuyển động

a) Các hàm và thủ tục tạo ảnh chuyển động

Một ảnh được lưu trữ trong bộ nhớ bằng tập hợp các bit (gọi là ảnh bit, bit image), chẳng hạn với ảnh 16 màu thì mỗi pixel tương ứng với 4 bit (ứng với một màu từ 0, 1, 2, . . ., 15). Việc xử lý ảnh bit được tiến hành theo từng bit.

Các hàm và thủ tục sau dùng để tạo ảnh chuyển động:

• **Hàm ImageSize(x1, y1, x2, y2: integer): word** xác định số byte cần thiết cho thủ tục GetImage lưu vùng chữ nhật trên màn hình (x1, y1, x2, y2). Kích thước ảnh bao gồm thêm cả ba word: word đầu lưu độ rộng của vùng chữ nhật, word thứ hai lưu chiều cao vùng, word thứ ba dùng để dự trữ. Máy lưu các điểm ảnh theo hàng, mỗi

điểm ảnh chiếm 4 bit, nếu số lượng [(số điểm ảnh một hàng * 4 bit) / 8 bit] byte không chia hết cho 4 thì máy sẽ thêm từ 1 đến 3 byte để số byte cần lưu một hàng chia hết cho 4. Ví dụ nếu vùng ảnh là (1,1,100,100) thì số byte cần thiết để lưu vùng ảnh là [(100 điểm * 4 bit / 8 bit + 2 byte] * 100 hàng + 6 byte = 5206 byte. Nếu bộ nhớ yêu cầu để lưu vùng lớn hơn 64 KB thì hàm trả về giá trị 0 và GraphResult = -11.

• **Thủ tục GetImage(x1, y1, x2, y2: integer; var BitMap)** chép ảnh bit nằm trong vùng chữ nhật (x1, y1, x2, y2) vào biến BitMap. BitMap là một tham số không định kiểu. Hai word đầu tiên của BitMap lưu trữ chiều rộng và chiều cao của vùng, word thứ ba dùng để dự trữ, phần còn lại của BitMap được dùng để ghi ảnh bit. Dùng hàm ImageSize để xác định yêu cầu về kích thước của BitMap.

• **Thủ tục PutImage(x,y: integer; var BitMap ; BitBlt: word)**: đặt ảnh bit lưu trong BitMap lên màn hình, góc trên bên trái của vùng ảnh có toạ độ là (x, y). BitMap là tham số không định kiểu chứa chiều cao và chiều rộng của vùng ảnh, và ảnh bit sẽ được đặt lên màn hình. Tham số BitBlt xác định phép toán thao tác bit nào sẽ được dùng để đặt ảnh bit lên màn hình. Mỗi một hằng số tương ứng với một phép toán thao tác bit:

Tên hằng	Giá trị	Phép toán bit
CopyPut, NormalPut	0	MOV
XORPut	1	XOR
ORPut	2	OR
ANDPut	3	AND
NOTPut	4	NOT

Ý nghĩa các phép toán thao tác bit:

Bit A (ảnh)	Bit B (nền)	A and B	A or B	C=A xor B	A xor C	not A
1	1	1	1	0	1	0
1	0	0	1	1	0	
0	1	0	1	1	1	1
0	0	0	0	0	0	

Ví dụ, PutImage(x, y, BitMap, NormalPut) sẽ đặt ảnh lưu trong Bitmap tại (x,y) bằng cách dùng lệnh MOV của ngôn ngữ Assembler với từng byte trong ảnh (ảnh mới sẽ ghi đè lên ảnh cũ trên màn hình). Lệnh PutImage(x, y, BitMap, XorPut) sẽ đặt ảnh lưu trong BitMap tại (x,y) bằng cách dùng lệnh XOR của ngôn ngữ Assembler với từng byte trong ảnh, lệnh này thường dùng để tạo ảnh chuyển động trên màn hình có nền. Lệnh PutImage(x, y, Bitmap, NotPut): đảo các bit trong BitMap, sau đó đặt ảnh lưu trong Bitmap lên màn hình tại (x,y) bằng cách dùng lệnh MOV với từng byte trong ảnh, như vậy ảnh sẽ bị đảo so với ảnh gốc.

Từ cột 5 và cột 6 của bảng trên ta có nhận xét: khi in hai lần một ảnh vào cùng một vị trí theo cách XorPut thì ảnh sẽ bị xoá và nền cũ hiện lại (cột 6 trùng với cột 2). Ta sẽ ứng dụng nhận xét này để tạo ảnh chuyển động mà vẫn giữ nguyên màn hình nền. Từ cột 5 ta thấy: dùng cách XorPut để in ảnh thì ảnh bị đổi màu (xem chương trình XorPut.pas ở dưới), nếu ảnh vẽ và ảnh nền cùng màu thì màu kết quả theo cách XorPut là màu nền.

Chương trình **PutImag.pas**: lưu vùng ảnh chữ nhật vào P^, 21 lần in ảnh lưu ra màn hình theo cách CopyPut.

```
uses Graph;
var Gd, Gm,i: Integer; P: Pointer; Size: Word;
begin Gd := Detect; InitGraph(Gd, Gm, ' ');
if GraphResult <> grOk then   Halt(1);
Bar(0, 0, GetMaxX, GetMaxY);
Size := ImageSize(10, 20, 30, 40); GetMem(P, Size);
GetImage(10, 20, 30, 40, P^); Readln; ClearDevice;
for i:=0 to 20  do PutImage(10+i*22, 10+i*22, P^, CopyPut);
Readln; CloseGraph;
end.
```

Chương trình **XorPut.pas**: đặt nền màn hình xanh sáng, vẽ khung chữ nhật nét trắng, vẽ hình tròn tô xanh lơ ở trong khung, vẽ hình tròn tô xanh bên trong, rồi hình tròn trắng. Copy vùng chữ nhật vào P^: màu nền xanh sáng trong khung không được copy, máy chỉ ghi nhớ đó là màu thứ 0 (màu nền). Đặt lại màu nền là đen, vẽ một hình chữ nhật tô trắng, dùng lệnh PutImage với cách chép ảnh XorPut để in ảnh lưu trong P^ lên vùng chữ nhật, ảnh sẽ bị đổi màu: phần hình tròn màu xanh lơ biến thành đỏ sáng, phần hình tròn màu xanh biến thành màu vàng, phần hình tròn màu trắng và các nét vẽ màu trắng biến thành màu nền đen.

```
uses Graph,crt;
var Gd, Gm,i: Integer; P: Pointer; Size: Word;
begin Gd := Detect; InitGraph(Gd,Gm,'');DirectVideo:=False;
if GraphResult <> grOk then Halt(1);

setBkColor(9);       { nền Xanh sáng }
rectangle(0,0,200,200); readln;{ nét vẽ khung chữ nhật Trắng}
setFillstyle(1,3); FillEllipse(100,100,85,85);readln;   {tô Xanh lơ}
setFillstyle(1,1); Fillellipse(100,100,60,60);readln;   {tô Xanh}
setFillStyle(1,15); Fillellipse(100,100,40,40);readln; {tô Trắng}
Size := ImageSize(0,0,200,200);
GetMem(P, Size);
GetImage(0, 0, 200, 200, P^); {không ghi nền ảnh}
ClearDevice; writeln('Size= ',size); readln; {20910 byte}

setBkColor(0); setfillstyle(1,white);
bar(0,0,539,379);  readln;{ toàn màn hình vẽ bằng màu Trắng}
PutImage(120,100, P^, XorPut); readln;
{ 3 xor 15 = 12 đỏ sáng, 1 xor 15 = 14 vàng, 15 xor 15 = 0 màu nền đen}
Readln; CloseGraph;
end.
```

- **Thủ tục SetWriteMode(WriteMode: integer)**: đặt chế độ vẽ khi vẽ các đường bằng các thủ tục DrawPoly, Line, LineRel, LineTo và Rectangle. Tham số WriteMode nhận các giá trị hằng CopyPut, XorPut, OrPut, AndPut, NotPut như trong thủ tục PutImage. Mỗi hằng tương ứng với một phép toán bit giữa mỗi byte của đường thẳng và byte tương ứng trên màn hình. XORPut dùng lệnh XOR để tổ hợp đường thẳng với màn hình. Hai lệnh XOR liên tiếp sẽ xoá đường thẳng và khôi phục màn hình theo trạng thái cũ. CopyPut sẽ dùng lệnh MOV để đặt đường thẳng đè lên những gì đã có trên màn hình.

Chương trình **SetWriM.pas**: tạo màn hình nền là các chấm điểm, vẽ các khung chữ nhật nét to trên màn hình có toạ độ góc trên trái là ngẫu nhiên, khung chữ nhật

không xoá nền vì chế độ vẽ là XorPut.

```
uses Crt, Graph;
var Driver, Mode, I: Integer; X1, Y1, Dx, Dy: Integer;
    FillInfo: FillSettingsType;
begin DirectVideo := False;  Randomize;
Driver := Detect; InitGraph(Driver, Mode, ' ' );
if GraphResult < 0 then Halt(1); GetFillSettings(FillInfo);
SetFillStyle(WideDotFill, FillInfo.Color);
Bar(0,0,GetMaxX,GetMaxY); Dx:=GetMaxX div 4; Dy:=GetMaxY div 4;
SetLineStyle(SolidLn, 0, ThickWidth); SetWriteMode(XORPut);
repeat X1 := Random(GetMaxX - Dx); Y1 := Random(GetMaxY - Dy);
Rectangle(X1, Y1, X1 + Dx, Y1 + Dy); Delay(1000);
Rectangle(X1, Y1, X1 + Dx, Y1 + Dy);
until KeyPressed; Readln; CloseGraph;
end.
```

b) Tạo ảnh chuyển động không có màn hình nền. Phương pháp này chỉ sử dụng trong trường hợp vật chuyển động qua vùng màn hình trơn (không có ảnh các vật thể khác, chỉ có màu nền). Các bước thực hiện: tạo ảnh bằng màu vẽ, xoá ảnh bằng màu nền, tạo lại ảnh bằng màu vẽ tại vị trí mới

Chương trình **Bong.pas**: vẽ một quả bóng màu đỏ trên nền Cyan, khi ấn một phím mũi tên quả bóng liên tục chuyển động theo hướng mũi tên cho đến biên màn hình, ấn phím ESC để dừng chương trình.

```
uses crt,graph;
const r=50; s=3;
var gd,gm,x,y,mx,my,nen: integer; c: char;

procedure Bong(x,y,mau: word);
begin setcolor(mau); setfillstyle(1,mau); fillellipse(x,y,r,r);
end;

begin gd:= detect; initgraph(gd,gm,'');
nen:=cyan; setbkcolor(nen); mx:= getmaxx; my:= getmaxy;
randomize; x:= random(mx-2*r)+r; y:=random(my-2*r)+r;
bong(x,y,red);

repeat  c := readkey;
if (c = #0) then
  begin c := readkey;
    case c of
     #75: while (not keypressed) and (x>r) do
          begin bong(x,y,nen);x:=x-s;bong(x,y,red);delay(100) end;
     #77: while (not keypressed) and (x<mx-r) do
          begin bong(x,y,nen);x:=x+s;bong(x,y,red);delay(100) end;
     #72: while (not keypressed) and (y>r) do
          begin bong(x,y,nen);y:=y-s;bong(x,y,red);delay(100) end;
     #80: while (not keypressed) and (y<my-r) do
          begin bong(x,y,nen);y:=y+s;bong(x,y,red);delay(100) end;
    end;
 end;
until c=#27; closegraph;
end.
```

c) Tạo ảnh chuyển động trên một màn hình nền

Thuật toán tạo ảnh chuyển động mà không xoá nền:

1. Vẽ ảnh trong vùng (x1, y1, x2, y2).

2. Tính số byte cần thiết để lưu ảnh thuộc vùng chữ nhật nhờ hàm ImageSize (giả sử là n), cấp phát vùng nhớ n byte và cho con trỏ P trỏ vào byte đầu tiên của vùng nhớ này bằng lệnh GetMem.

3. Dùng lệnh GetImage để chép ảnh từ vùng chữ nhật (x1, y1, x2, y2) vào vùng nhớ vừa cấp phát (ứng với biến động P^),

4. Xoá màn hình, tạo màn hình nền hoàn toàn mới,

5. Dùng lệnh PutImage in ảnh lưu trong P^ ra màn hình tại toạ độ (x,y) theo cách XorPut, dừng chương trình một lát, thay đổi các thành phần x,y để được điểm (x,y) mới, dùng PutImage in lại ảnh vẫn tại toạ độ (x,y) cũ theo cách XorPut để xoá ảnh và khôi phục nền cũ, dùng PutImage in ảnh ở điểm (x,y) mới . . . Thời gian giữa xoá ảnh cũ và in ảnh mới càng ngắn càng tốt.

Chương trình **CDong.pas**: vẽ một viên đạn nghiêng sang phải 45 độ, ghi ảnh vào biến con trỏ P, đặt nền màn hình màu Cyan, viết 2 dòng chữ màu Blue ở giữa màn hình, vẽ một mặt trăng màu trắng ở giữa nửa trên màn hình. Cho 4 viên đạn xuất phát ngẫu nhiên từ cạnh trái màn hình và từ cạnh đáy màn hình chuyển động từ phía trái sang phía phải với góc nghiêng 45 độ.

```
uses graph,crt;
var gd,gm,x1,y1,x2,y2,x3,y3,x4,y4,u1,u2,u3,u4,v1,v2,v3,v4:integer;
    p: pointer;
begin gd:=VGA; gm:=VgaHi; initgraph(gd,gm,'');
      setcolor(red); setfillstyle(1,red);
line(0,12,8,20); line(8,20,18,10); line(18,10,20,0);
line(20,0,10,2); line(10,2,0,12);  floodfill(10,10,red); readln;
getmem(p,imagesize(0,0,20,20)); getimage(0,0,20,20,p^); cleardevice;

setbkcolor(cyan); settextstyle(0,0,2); settextjustify(1,1);
setcolor(blue); outtextxy(320,240,'Bui The Tam, Lop 12 A');
outtextxy(320,280,'Truong Trung hoc Pho thong Ly Bi');

setcolor(white); ellipse(320,100,180,360,50,50);
ellipse(320,100,180,360,50,25); setfillstyle(1,white);
floodfill(320,130,white);

x1:=0; y1:=random(460); x2:=random(620); y2:=479;
x3:=0; y3:=random(460); x4:=random(620); y4:=479;
repeat  putimage(x1,y1,p^,1); putimage(x2,y2,p^,1);
        putimage(x3,y3,p^,1); putimage(x4,y4,p^,1);
u1:=x1; v1:=y1;u2:=x2; v2:=y2; u3:=x3; v3:=y3;u4:=x4; v4:=y4;
x1:=x1+5; y1:=y1-5; if y1<0 then begin x1:=0; y1:=random(480) end;
x2:=x2+5; y2:=y2-5;
if (y2<0) or(x2>640-20) then begin x2:=random(620); y2:=479 end;
x3:=x3+5; y3:=y3-5; if y3<0 then begin x3:=0; y3:=random(480) end;
x4:=x4+5; y4:=y4-5;
if (y4<0) or(x4>640-20) then begin x4:=random(620); y4:=479 end;
delay(20); putimage(u1,v1,p^,1); putimage(u2,v2,p^,1);
           putimage(u3,v3,p^,1); putimage(u4,v4,p^,1);
```

```
until keypressed; closegraph; dispose(p);
end.
```

Việc chép ảnh bằng lệnh PutImage theo cách XorPut có nhược điểm là ảnh bị đổi màu khi đi qua ảnh nền. Để khắc phục điều này ta có thể kết hợp việc vẽ ảnh với việc cất ảnh nền ở vị trí định vẽ để sau đó có thể khôi phục ảnh nền cũ bằng CopyPut. Chương trình **LuuNen.pas** minh hoạ cách làm này: tạo 500 hình tròn có màu ngẫu nhiên làm ảnh nền, cho quả bóng màu đỏ chuyển động ngang màn hình mà không xoá nền.

```
uses graph,crt;
var i,k,gd,gm,x1,y1,u1,v1,r:integer; n: word; p: pointer;
begin gd:=VGA; gm:=VgaHi; initgraph(gd,gm,''); randomize; r:=50;
for i:=1 to 500 do
  begin k:= random(15)+1; setcolor(k); setfillstyle(1,k);
        Fillellipse(random(640-40)+20,random(480-40)+20,20,20);
  end;

x1:=r; y1:=240; n:=imagesize(x1-r,y1-r,x1+r,y1+r); getmem(p,n);
repeat GetImage(x1-r,y1-r,x1+r,y1+r,p^);
setcolor(LightRed); setfillstyle(1,LightRed); FillEllipse(x1,y1,r,r);
u1:=x1; v1:=y1; x1:=x1+5;
if x1>639-r then
      begin delay(500); x1:=r; putimage(u1-r,v1-r,p^,0) end
else  begin delay(500); putimage(u1-r,v1-r,p^,0); end;
until keypressed; dispose(p); closegraph;
end.
```

d) Vẽ ảnh chuyển động nhờ phép lật trang màn hình

Hình ảnh đồ hoạ có thể lưu vào các vùng khác nhau trên RAM màn hình, mỗi vùng gọi là một trang màn hình. Do việc chuyển đổi giữa các trang màn hình được thực hiện dễ dàng nên có thể tạo các hình ảnh chuyển động bằng cách luân phiên vẽ và lật trang.

• **Thủ tục SetActivePage(Page: word)**: đặt hoạt động cho trang có số hiệu Page, chưa hiện trang này (thường được gọi trước khi thực hiện các lệnh xử lý đồ hoạ cho trang sắp được xem).

• **Thủ tục SetVisualPage(Page: word)**: hiện trang màn hình có số hiệu Page (đã được chuẩn bị trước bằng lệnh SetActivePage).

Chương trình **LatTrang.pas**: vẽ đôi mắt nhìn qua lại trên màn hình.

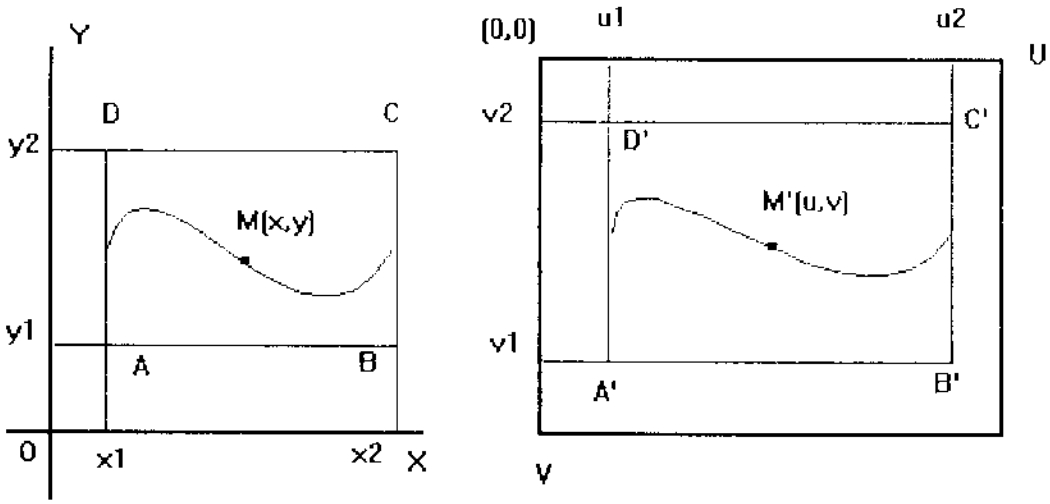
```
uses Crt, Graph;
var Gd,Gm,x,y: integer; Page:Word;

procedure DrawEye(Mode: word);
var  DeltaX : integer;
begin SetColor(yellow); SetFillStyle(1,yellow);
FillEllipse(getmaxx div 2, getmaxy div 2,100,100); {vẽ mặt }
SetColor(red); SetFillStyle(1,red);
FillEllipse(getmaxx div 2, getmaxy div 2+70,25,9);   { vẽ miệng }
SetColor(White); SetFillStyle(1,White);
Sector(x,y,0,360,20,10); Sector(x+100,y,0,360,20,10); { vẽ hai mắt }
SetFillStyle(1,Blue); SetColor(Blue); { vẽ hai con ngươi }
```

```
case Mode of
      0: DeltaX := -5;
      1: DeltaX :=  5;
end;
Sector(x+DeltaX,y,0,360,10,10);
Sector(x+DeltaX+100,y,0,360,10,10);
end;

begin Gd := VGA; Gm := VGAMed;
InitGraph(Gd,Gm,''); if GraphResult <> grOk then Halt(1);
x := GetMaxX div 2 - 50;
y := GetMaxY div 2; Page := 0;
while not KeyPressed do
  begin SetActivePage(Page); DrawEye(Page);
        SetVisualPage(Page);
        if Page=0 then Page:=1 else Page:=0; Delay(500);
  end; CloseGraph;
end.
```

8. Vẽ đồ thị các hàm số

a) Chuyển một hình vẽ trong toạ độ Đề các lên màn hình

Trong toán học ta vẽ đồ thị của hàm số y = F(x) trong toạ độ Đề các với trục X hướng sang phải, trục Y hướng lên trên, thang chia các trục do ta tự chọn. Hệ toạ độ (U,V) trên màn hình lại có gốc toạ độ ở góc trên trái màn hình, trục U hướng sang phải và có giá trị cố định từ 0 đến 639, trục V hướng xuống dưới và có giá trị từ 0 đến 479.



Do mặt phẳng vẽ của hệ toạ độ (x,y) là vô hạn và mặt phẳng vẽ của hệ toạ độ (u,v) là hữu hạn nên tại một thời điểm ta chỉ có thể giới hạn xem được một vùng chữ nhật ABCD trên hệ toạ độ (x,y), toạ độ của điểm A là (x1,y1) và của điểm C là (x2,y2). Vấn đề là ta cần phải ánh xạ các điểm vẽ trong hình chữ nhật ABCD vào một vùng chữ nhật A'B'C'D' trên màn hình, toạ độ của điểm A' là (u1,v1) và của điểm C' là (u2,v2). Phép ánh xạ này phải đảm bảo chuyển các điểm A, B, C, D thành các điểm A', B', C', D' tương ứng, hình ảnh trong hệ toạ độ (x,y) khi chuyển sang hệ toạ độ (u,v) có thể bị co dãn..

Giả sử điểm vẽ M có toạ độ (x,y), qua phép ánh xạ chuyển thành điểm M' có toạ

độ (u,v). Các giá trị u và v được tính theo công thức:

```
Tlx = (u2-u1) / (x2-x1); Tly = (v1-v2) / (y2-y1)
u = u1 + Round((x-x1)*Tlx); v = v1 - Round((y-y1)*Tly)
```

Tlx, Tly là tỷ lệ co dãn theo trục X và Y tương ứng. Hàm Round(z: real) làm tròn số thực x thành số nguyên gần nhất.

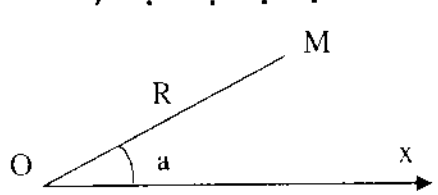
Thuật toán để vẽ đồ thị hàm số y = F(x) bất kỳ: vẽ đồ thị hàm số trong hệ toạ độ (x,y), giới hạn vùng nhìn ABCD, giới hạn vùng ảnh trên màn hình A'B'C'D', dùng ánh xạ trên để chuyển các điểm trên đồ thị (x,y) thành các điểm ảnh (u,v) trên màn hình.

Chương trình **DoThi.pas**: vẽ đồ thị hàm số Y = 0.5 * x * Sin(x) với vùng vẽ là x1= -50, y1= -20, x2= 50, y2= 20 và vùng hiển thị trên màn hình là u1= 50, v1= 429, u2= 625, v2= 10.

```
uses crt, graph;
var gd,gm,u,v,u1,u2,v1,v2: integer;
    x,y,x1,y1,x2,y2,tlx,tly: real; s1,s2,s3,s4: string[5];
function F(x:real): real;
begin  f:=0.5*x*sin(x);  end;

procedure Doi(x,y: real; var u,v: integer);
begin  u:=u1+round((x-x1)*tlx); v:=v1-round((y-y1)*tly) end;

begin clrscr; x1:= -50; y1:= -20; x2:= 50; y2:= 20;
u1:= 50; v1:= 429; u2:= 625; v2:= 10;
tlx:=(u2-u1)/(x2-x1); tly:=(v1-v2)/(y2-y1);
gd:=detect; initgraph(gd,gm,'');
setcolor(red); rectangle(u1,v2,u2,v1);

setcolor(cyan); settextstyle(0,0,1);
settextjustify(1,2); str(x1:3:0,s1); outtextxy(u1,v1+8,s1);
str(x2:3:0,s2);outtextxy(u2,v1+8,s2);
settextjustify(2,1); str(y1:3:0,s3); outtextxy(u1-5,v1,s3);
str(y2:3:0,s4);outtextxy(u1-5,v2,s4);
settextjustify(1,1);  setcolor(lightmagenta);
outtextxy(320,470,'Bui The Tam, lop 11A, Truong THPT Ly Nam De');
x:=x1;
repeat y:=f(x); Doi(x,y,u,v);
if (v>=v2) and(v<=v1) then Putpixel(u,v,white); x:=x+0.005;
until (x>x2);   readln;
end.
```

Phương pháp trên có thể áp dụng để vẽ các đường cong cho dưới dạng tham số: x= f(t), y = h(t) với t biến thiên trong khoảng a<=t<=b. Đầu tiên cho t_i=a, với mỗi t_i ta tính (x_i, y_i) và chuyển sang toạ độ màn hình để vẽ điểm này, tiếp theo $t_{i+1}=t_i+\varepsilon$ và vẽ điểm i+1. . . Ví dụ đường xoắn dạng lò xo có phương trình x = t + cos(3t), y = t + sin(3t).

b) Hệ toạ độ cực



Toạ độ cực gồm một điểm gốc O, một tia Ox. Một điểm M trong mặt phẳng xác định bởi cặp số (R,a), trong đó R là độ dài đoạn MO, a là góc tạo bởi MO với trục Ox. Nếu góc a trong phương trình R = F(a) nằm ở

dạng lượng giác hay bội số thì hàm sẽ tuần hoàn. Ví dụ phương trình:

$$R = 1 - Cos^2(5*a*\pi/180)$$

với $a \in [0^0, 360^0]$ sẽ tạo nên bông hoa 10 cánh. Một điểm M = (R, a) của toạ độ cực có thể đổi sang toạ độ Đề các (x,y) theo công thức: x = R Cos(a), y = R Sin(a).

Chương trình **Hoa10.pas** nhằm vẽ đồ thị trên trong toạ độ cực:

```
uses crt, graph;
var gd, gm, u, v, u1, u2, v1, v2: integer;
    x, y, x1, y1, x2, y2, tlx, tly, t, t1, r: real;
begin clrscr;
x1:=-1; x2:=1; y1:=-1; y2:=1;
u1:=30; u2:=475; v1:=450; v2:=5;
tlx:=(u2-u1)/(x2-x1); tly:=(v1-v2)/(y2-y1);
gd:=detect; initgraph(gd,gm,'');
setcolor(red);
rectangle(u1,v2,u2,v1); t:=0; t1:=pi/180;
repeat
r:= 1-SQR(cos(5*t*t1));
x:= r* cos(t * t1); y:= r * sin(t * t1);
u:= u1 + Round((x-x1) * tlx);
v:= v1 - Round((y-y1) * tly);
if (u>=u1) and (u<=u2) and (v>=v2) and(v<=v1) then
Putpixel(u,v,white);
t:=t+0.05;
until (t> 360); readln;
end.
```

9. Đồ hoạ trong chế độ 256 màu

Để có thể sử dụng 256 màu trong chế độ đồ hoạ trên đĩa cần có tệp SVGA256.BGI. Chương trình **Mau256.pas** minh hoạ cách dùng 256 màu.

```
uses Graph, Crt;
var  gd,gm,x,y,k:integer; s: string;
BEGIN
gd := InstallUserDriver('Svga256',nil);                    {1}
gm:=2;                                                     {2}
if gd = grError then Halt;
InitGraph(gd, gm,'');
if GraphResult <> 0 then
   begin
      Writeln('Graphics error ',GraphResult,' : ',GraphErrorMsg(GraphResult)) ;
      Halt; end;
      { Vẽ 256 vạch màu  với các màu khác nhau, mỗi vạch rộng 2 cột điểm}
for k := 0 to 255 do
        begin  for y:=0 to getmaxy do PutPixel(2*k,y,k);
               for y:=0 to getmaxy do PutPixel(2*k+1,y,k);
        end;
                    { Từ cột 511 đến cột 639 vẽ màu trắng }
for x:=255*2+1 to 639 do for y:=0 to getmaxy do PutPixel(x,y,15);
readln; cleardevice;
      { Vẽ 255 hình tròn có tô màu với các màu từ 1 đến 255 ở giữa màn hình.
```

góc trên bên trái màn hình thông báo số hiệu của màu đang dùng }

```
settextstyle(0,0,6); settextjustify(0,2);
for x:=1 to 255 do
 begin  str(x,s);  setcolor(x);OutTextXY(0,10,s);
        SetfillStyle(solidfill,x);
        FillEllipse(320,240,238,238); delay(500); cleardevice;
 end;
CloseGraph;
END.
```

Lệnh (1) trong chương trình nhằm cài đặt chương trình điều khiển màn hình ở chế độ 256 màu Svga256.bgi. Lệnh (2) đặt màn hình ở chế độ 640 cột và 480 hàng điểm. Biến Gm trong (2) còn có thể nhận các giá trị: 0 (ứng với màn hình ở chế độ 320*200 điểm), 1 (640*400 điểm), 3 (800*600 điểm), 4 (1024*768 điểm).

Bài tập

1. Lập ba thủ tục: in một xâu ký tự trên màn hình đồ hoạ nhưng con trỏ vẽ không xuống dòng, in một xâu ký tự và con trỏ vẽ xuống dòng, đọc một xâu ký tự từ bàn phím. Dùng ba thủ tục này để nhập họ tên, số ngày công, lương ngày của một người. In ra màn hình đồ hoạ họ tên và lương tháng.

2. Tạo một bút vẽ là một điểm sáng màu vàng ở giữa màn hình đồ hoạ, dùng 4 phím mũi tên để di chuyển bút vẽ và vẽ một hình tuỳ ý trên màn hình.

3. Hãy vẽ liên tục các hình chữ nhật nằm trọn trong màn hình cho đến khi ấn một phím bất kỳ thì kết thúc. Yêu cầu: chiều dài, chiều rộng, màu nét vẽ viền, mẫu tô và màu tô trong ruột hình chữ nhật đều là ngẫu nhiên, độ dài tối thiểu của một cạnh hình chữ nhật là 10 điểm.

4. Vẽ quả bóng màu đỏ và ghi vào một biến con trỏ. Xoá màn hình, vẽ bầu trời đầy sao, cho quả bóng chuyển động ngẫu nhiên (hướng ngẫu nhiên, độ dài bước ngẫu nhiên không quá 5 điểm) trên màn hình mà không xoá các ngôi sao.

5. Vẽ đồ thị hàm số Y = sin(x)*exp(-0.1*x) (mô tả một dao động tắt dần) với các vùng vẽ như sau: x1= -35, y1= -20, x2=15, y2= 20, u1= 50, v1= 429, u2= 625, v2= 10.

6. Vẽ đồ thị hàm số f(x) trong hệ toạ độ Đề các với hai trục toạ độ có chia đơn vị (kèm theo số). Chương trình cho phép người dùng nhập vào vùng vẽ mới (x1, y1, x2, y2) trên một dòng ở đáy màn hình để phóng to hay thu nhỏ đồ thị, vẽ lại đồ thị theo vùng vẽ mới, khi nào nhập vào chữ "KT" thì kết thúc chương trình.

7. Vẽ đường xoáy trôn ốc ngược chiều kim đồng hồ trong toạ độ cực cho bởi R= 50 - 0.3* a* PI/ 180, trong đó a là góc tăng từ 0 độ cho tới khi R < 0 thì dừng, mỗi lần tăng 0.05 độ. Vùng giới hạn trong hệ toạ độ (x,y) là x1= -51, x2= 51, y1= -51, y2= 51. Vùng giới hạn trên màn hình là u1= 30, u2= 475, v1= 450, v2= 5.

8. Vẽ các đường cong sau trong toạ độ cực:

a) R = 1 + cos(a) , 0 <= a <= 2π (đường hình tim)

b) R = 5 - | sin(10a) | , 0 <= a <= 2π (đường hình bánh răng)

c) R = sqrt(π / a), a > 0 (đường xoắn hình sên).

9. Vẽ các đường cong sau cho dưới dạng tham số:

a) x = t + cos(3t), y = t + sin(3t), -10 <= t <= 10 (đường xoắn lò so).

b) $x = \cos^3(t)$, $y = \sin^3(t)$, 0 <= t <= 2π (đường hình sao)

c) x = 2 cos(t) - cos(2t), y = 2 sin(t) - sin(2t), 0 <= t <= 2π (đường hình tim).

10. Lập trình vẽ một chiếc đồng hồ chạy theo thời gian lưu trong máy tính.

11. Lập trình vẽ một quả bóng chuyển động thẳng trong một hình chữ nhật theo nguyên tắc: bóng luôn chuyển động thẳng theo một góc nghiêng 45 độ đối với cạnh sắp tới, khi bóng chạm phải cạnh hình chữ nhật thì nảy trở lại theo góc nghiêng 45 độ và tiếp tục chuyển động thẳng.

12. Tạo một menu dọc có 5 chức năng trong chế độ đồ hoạ (chọn chức năng thứ năm thì kết thúc chương trình), khi chọn một chức năng thì thực hiện một thủ tục đồ hoạ, thực hiện xong ấn Enter lại trở về menu.

13. Cho màn hình màu CYAN, vẽ một chiếc gương màu trắng hình bình hành ở chính giữa gần cạnh đáy màn hình. Vẽ một chùm ánh sáng màu đỏ gồm 3 tia sáng xuất phát từ một chiếc đèn, chùm sáng chiếu từ bên phải màn hình vào gương theo một góc chiếu và phản xạ sang phía trái màn hình. Hãy vẽ chùm tia sáng thay đổi khi góc chiếu của chùm sáng vào gương tăng lên (ấn phím T), giảm xuống (ấn phím G), khi ấn phím X thì kết thúc chương trình.

14. Hãy tìm bao lồi của n điểm cho trước trên mặt phẳng (x1, y1), (x2, y2), ... (xn, yn). Vẽ bao lồi tìm được trên màn hình đồ hoạ.

Chương 10

Unit tự tạo

1. Unit tự tạo

Ngoài các thường trình có sẵn trong unit chuẩn của Turbo Pascal như System, Crt, Dos, Graph ... người sử dụng còn có thể xây dựng cho mình các thường trình bằng cách tự tạo các unit. Mục đích của unit tự tạo: tạo các thư viện hàm và thủ tục để tránh lập lại những công việc giống nhau, giải quyết vấn đề thiếu bộ nhớ khi thiết kế các chương trình lớn, tạo công cụ để liên kết các module chương trình với nhau mà vẫn đảm bảo tính độc lập cho từng lập trình viên.

a) Cấu trúc của một Unit

Cũng giống như bất kỳ một chương trình nguồn Pascal nào, tệp unit có dạng text và chứa các dòng lệnh của ngôn ngữ Pascal. Tuy nhiên chúng khác nhau về cấu trúc. Các thành phần của unit bao gồm: phần định nghĩa (tên unit), phần giao tiếp (Interface), phần cài đặt (Implementation), phần khởi tạo (Initialization), phần kết thúc (End).

• **Phần định nghĩa** nhằm mục đích báo cho trình biên dịch biết đây là unit (nếu không có phần này thì mặc nhiên cho là chương trình). Phần này chỉ cần một câu lệnh: `Unit TenUnit ;` trong đó TenUnit là tên của Unit, nó phải trùng với phần chính của tên tệp lưu trên đĩa. Ví dụ nếu tạo một unit có tên là Tools lưu trên đĩa bởi tệp Tools.pas thì ở dòng định nghĩa phải là `Unit Tools ;`

• **Phần giao tiếp** khai báo tất cả những gì dùng chung (các chương trình và Unit khác có thể sử dụng). Phần này bắt đầu bằng từ khoá Interface, theo sau là tên các Unit, hằng, biến, kiểu dữ liệu, tên hàm và thủ tục dùng để giao tiếp với bên ngoài.

• **Phần cài đặt** chứa những gì dành riêng cho Unit và dùng để cài đặt các thủ tục, hàm đã khai báo ở phần giao tiếp. Đây chính là phần cốt lõi, quyết định chức năng của các thường trình dùng chung. Phần này bắt đầu bằng từ khoá Implementation. Các chương trình con thuộc phần này được tự do sử dụng các khai báo trong phần Interface, ngoài ra mỗi chương trình con có thể có tập các biến riêng của mình. Trong thân một chương trình con có thể gọi tới các chương trình con khác của Unit.

• **Phần khởi tạo** (có thể có hoặc không): phần này nằm ngay trước phần kết thúc, bắt đầu bằng từ khoá Begin, theo sau là các lệnh để khởi tạo. Nếu trong một chương trình chính có dùng nhiều Unit thì phần khởi tạo (nếu có) của từng unit sẽ được thực hiện (theo thứ tự chỉ ra trong Uses) trước khi thực hiện chương trình.

• **Phần kết thúc** chỉ gồm từ khoá **End.** (có dấu chấm ở sau) báo cho trình biên dịch biết đã hết phần cài đặt của unit. Tương tự như tệp chương trình nguồn, tất cả những gì viết sau từ khoá End. đều không có tác dụng.

Dạng mẫu của một Unit (tên tệp là MyUnit.pas):

Unit MyUnit ;
Interface
    Khai báo Uses

Khai báo Const, Type, Var
Khai báo Procedure, Function

Implementation

Khai báo Uses
Khai báo Const, Type, Var
Cài đặt các Procedure, Function

Begin

Các lệnh khởi tạo - chỉ xuất hiện khi có từ khoá Begin

End.

b) Cách sử dụng Unit

Sau khi tạo được unit, có thể biên dịch vào đĩa hoặc bộ nhớ. Đối với các unit thường sử dụng nên biên dịch ra đĩa thành các tệp *.TPU tương ứng bằng cách: khởi động Pascal, nạp tệp unit cần dịch, vào menu Compile, chọn Destination là Disk, ấn Alt+F9. Các chương trình muốn sử dụng unit nào, chỉ cần thêm tên unit đó vào câu lệnh Uses

Tệp **TVien.pas** sau nhằm tạo một unit của người dùng có tên là TVien:

```
Unit TVien;

Interface
const nmax=100;
type mang1= array[1..nmax] of real;
     mang2= array[1..nmax,1..nmax] of integer;
var i,j,n,m: integer;
function TichVH(n:integer; x,y: mang1): real;
procedure ChuyenVi(n: integer; var P: mang2);

Implementation
function TichVH;
var i: integer; s: real;
begin  s:=0;
       for i:=1 to n do s:=s+x[i]*y[i];
       TichVH:= s;
end;

procedure ChuyenVi;
var i,j,k: integer;
begin  for i:=1 to n-1 do
         for j:=i+1 to n do
           begin  k:=P[i,j];
                  P[i,j]:=P[j,i]; P[j,i]:=k
           end;
end;
begin  n:= 5;
end.
```

Các chương trình khác có thể dùng các thành phần sau của Unit TVien: hằng số nmax; các kiểu dữ liệu mang1, mang2; các biến nguyên i, j, n, m; hàm TichVH; thủ tục ChuyenVi. Trong phần khởi tạo: khởi tạo biến n bằng 5. Sau khi biên dịch thành tệp Tvien.tpu ra đĩa, ta có thể lập hai chương trình: tính tích vô hướng hai véc tơ, chuyển vị ma trận.

Chương trình **Tich2Vt.pas** tính tích vô hướng hai véc tơ nhập vào từ bàn phím:

```
uses crt,TVien;
var  a,b: mang1;
begin
clrscr;
for i:=1 to n do
    begin write('a',i,', b',i,' = ');
          readln(a[i],b[i])
    end;
writeln('Tich vo huong = ',TichVH(n,a,b):1:5);
readln;
end.
```

Trong chương trình này không phải khai lại kiểu Mang1, các biến nguyên i, n cũng không cần khai, giá trị của n bằng 5 (lấy từ Unit TVien). Như vậy mỗi véc tơ a, b đều có 5 phần tử.

Chương trình **ChuyenVi.pas** nhằm chuyển vị ma trận vuông A cấp m qua đường chéo chính.

```
uses crt,TVien;
var a:mang2;
begin clrscr;
write('Co ma tran m = '); readln(m);
for i:=1 to m do
   for j:=1 to m do
         begin write('a[',i,',',j,'] = ');
               readln(a[i,j])
         end;
for i:=1 to m do
  begin for j:=1 to m do write(a[i,j],'   ');
        writeln
  end;
readln; chuyenvi(m,a);
for i:=1 to m do
     begin for j:=1 to m do write(a[i,j],'   ');
           writeln
     end;
readln;
end.
```

Trong chương trình này không phải khai lại kiểu Mang2, các biến nguyên i, j, m cũng không cần khai.

Chú ý :

• Thư mục chứa các tệp TPU được xác định như sau: trong môi trường phát triển tích hợp (IDE) vào menu Options, chọn Directories, nhập đường dẫn tới thư mục TPU trong hộp "EXE & TPU Directories". Nếu để trống mục này, trình biên dịch sẽ tìm các tệp TPU trên thư mục hiện hành.

• Nếu một chương trình sử dụng các unit có chứa hàm hoặc thủ tục trùng tên thì tên unit khai báo sau trong câu lệnh USES sẽ được ưu tiên. Ví dụ: trong unit Crt chuẩn đã có thủ tục ClrScr; trong unit TVien ta tự xây dựng một thủ tục ClrScr (tạo một cửa sổ nhỏ trên màn hình). Giả sử ở đầu chương trình chính ta dùng lệnh

Uses CRT, TVIEN ;

Khi đó nếu trong chương trình chính dùng lệnh ClrScr thì máy sẽ thực hiện thủ tục ClrScr của Unit TVien. Muốn gọi đúng thủ tục ClrScr của unit Crt ta thực hiện như sau: Crt.ClrScr.

• Nếu chương trình chính và các Unit (hoặc giữa các Unit) có các đại lượng trùng tên, thì trong chương trình chính tên sẽ chỉ định đại lượng định nghĩa trong chương trình chính, trong unit nào,thì tên sẽ chỉ định đại lượng định nghĩa trong unit đó.

• Các unit có thể tham khảo vòng, ví dụ unit A dùng unit B, unit B dùng unit C, unit C lại dùng unit A. Điều này được thực hiện bằng cách chèn câu lệnh USES và tên các unit cần tham khảo vòng ngay sau khai báo Implementation.

2. Sử dụng tệp Include

Ngoài cách tổ chức thư viện theo Unit, Turbo Pascal còn có cách tổ chức thư viện chương trình theo các tệp Include (bao gồm). Giả sử chương trình nguồn của một hoặc nhiều chương trình con đã được thử nghiệm hoàn toàn chính xác, ta để chúng trên một tệp có đuôi PAS (ví dụ ThuVien1.pas), tệp này gọi là tệp Include. Sau này khi lập một chương trình khác, ta muốn sử dụng các chương trình con trong tệp ThuVien1.pas thì chỉ cần đánh dấu vị trí trong chương trình mà tại đó tệp Include sẽ được chèn vào bằng việc dùng dẫn hướng biên dịch: **{ $I ThuVien1.pas }**. Khi dịch chương trình gặp lệnh trên, máy sẽ lôi tệp nguồn ThuVien1.pas từ đĩa chèn vào vị trí đã đặt lệnh và biên dịch. So với cách dùng unit, cách này có thời gian dịch chương trình lâu hơn nhưng cách dùng lại rất đơn giản.

Giả sử tệp **ThuVien1.pas** có dạng:

```
Function CVDT(r: real): real;
Begin  CVDT := 2*Pi*r ; End;
```

Chương trình chính **DTron.pas** tính chu vi đường tròn:

```
var bk: real;
{ $I ThuVien1.pas }
begin
write('Vao ban kinh duong tron : '); readln(bk);
writeln('Chu vi duong tron : ', CVDT(bk):1:5);
readln;
end.
```

3. Các Unit Overlay

Overlay là các phần chương trình dùng chung bộ nhớ. Nhờ overlay ta có thể thực hiện các chương trình có kích thước lớn hơn kích thước vùng nhớ cho phép, vì từng phần của chương trình sẽ lần lượt thay nhau nằm thường trú trong bộ nhớ. Turbo Pascal quản lý Overlay ở cấp unit, đây là phần tử nhỏ nhất của chương trình có thể tạo thành các overlay. Khi chương trình có Overlay được biên dịch, Turbo Pascal sinh thêm tệp có đuôi OVR kèm theo tệp đuôi EXE (chương trình chỉ cần hai tệp này là chạy). Tệp EXE chứa phần tĩnh (không có Overlay) của chương trình và tệp OVR chứa tất cả các unit có Overlay.

Để hỗ trợ overlay, Turbo Pascal cung cấp một Unit chuẩn có tên là OVERLAY. Khi dùng overlay cần tuân theo các nguyên tắc:

- Bật dẫn hướng biên dịch Force Far Call cho chương trình và tất cả các Unit {$F+}.

- Đưa tên Unit Overlay vào vị trí đầu tiên sau mệnh đề USES của chương trình chính, sau đó là tên các unit khác mà chương trình cần dùng đến.

- Biên dịch các unit có Overlay và cả chương trình chính bằng dẫn hướng {$O+}.

- Liệt kê các unit có Overlay bằng dẫn hướng {$O Filename }, danh sách liệt kê viết sau mệnh đề USES của chương trình chính.

- Đảm bảo việc khởi động overlay phải được tiến hành trước bất kỳ câu lệnh nào.

Để giải thích cách dùng overlay ta xét hai ví dụ.

a) Các Unit bị Overlay không có phần khởi tạo

Khi các unit không có phần khởi tạo thì máy bắt đầu làm việc từ thân chương trình chính. Vì vậy để đảm bảo nguyên tắc cuối cùng ở trên thì câu lệnh đầu tiên trong thân chương trình chính phải là lời gọi đến thủ tục OvrInit().

Ví dụ minh hoạ gồm ba tệp:

• Tệp **Overl1.pas** để tạo unit Overl1, khi dịch tệp này ta được tệp Overl1.tpu. Unit này chứa thủ tục ThuTuc1 in một xâu ký tự.

```
{$O+,F+}
Unit Overl1;
Interface
Procedure ThuTuc1;

Implementation
Procedure ThuTuc1;
begin  writeln('Cong hoa xa hoi chu nghia Viet nam');
end;
end.
```

• Tệp **Overl2.pas** để tạo unit Overl2, khi dịch tệp này ta được tệp Overl2.tpu. Unit này chứa thủ tục ThuTuc2 in một xâu ký tự khác. Cả hai unit Overl1 và Overl2 gọi là các unit bị Overlay.

```
{$O+,F+}
Unit Overl2;
Interface
Procedure ThuTuc2;

Implementation
Procedure ThuTuc2;
begin  writeln('Doc Lap tu do hanh phuc');
end;
end.
```

• Tệp **Overtest.pas** là chương trình chính, nhằm gọi hai thủ tục ThuTuc1 và ThuTuc2. Khi biên dịch tệp này ta thu được hai tệp OverTest.exe và OverTest.ovr, các unit bị Overlay sẽ chuyển vào tệp OverTest.ovr.

```
{$O+,F+}
uses overlay, overl1, overl2;
{$O Overl1}
{$O Overl2}
begin
```

```
  OvrInit('overtest.ovr');
  ThuTuc1;
  ThuTuc2; readln;
end.
```

Giải thích:

- Trong tệp Overtest.pas sau câu lệnh Uses ta khai báo unit đầu tiên là Overlay. Dẫn hướng biên dịch {SO overl1} và {$O overl2} để khai báo các unit bị overlay.
- Cả ba tệp đều dùng dẫn hướng biên dịch {$O+, F+}.
- Câu lệnh đầu tiên của chương trình chính phải là lệnh khởi động trình quản lý overlay: `OvrInit('OverTest.ovr');`

b) Các Unit bị Overlay có phần khởi tạo

Khi các unit bị overlay có phần khởi tạo (ở cuối có phần Begin . . . End.) thì Turbo Pascal luôn luôn thực hiện phần khởi đầu trước khi thực hiện câu lệnh đầu tiên trong chương trình chính, nghĩa là các unit bị overlay sẽ thực hiện trước khi khởi động trình quản lý overlay (điều này vi phạm nguyên tắc cuối cùng đã nêu).

Để giải quyết vấn đề này, ta lập thêm một unit thứ ba là OvrStart (tên tệp là OvrStart.pas và được dịch thành OvrStart.tpu). Câu lệnh đầu tiên trong phần khởi tạo unit này chính là lời gọi tới thủ tục OvrInit(). Trong chương trình chính bỏ lệnh gọi thủ tục OvrInit(). Trong câu lệnh USES thuộc chương trình chính, unit OvrStart phải khai trước bất kỳ các unit bị overlay, điều này đảm bảo trình quản lý overlay sẽ được khởi động trước khi thực hiện bất kỳ unit bị overlay nào.

- Tệp Overl1.pas:

```
{$O+,F+}
Unit Overl1;
Interface
Procedure ThuTuc1;

Implementation
var S: string;
Procedure ThuTuc1;
begin  writeln(s);
end;
begin
      s:= 'Cong hao xa hoi chu nghia Viet nam';
end.
```

- Tệp Overl2.pas:

```
{$O+,F+}
Unit Overl2;
Interface
Procedure ThuTuc2;

Implementation
var s: string;
Procedure ThuTuc2;
begin writeln(s);
end;
begin
       s:= 'Doc Lap tu do hanh phuc';
end.
```

• Tệp OvrStart.pas:

```
  {$O+,F+}
Unit OvrStart;
Interface
uses Overlay;

Implementation
begin  OvrInit('OverTest.Ovr');
if OvrResult=OvrNotFound then
   begin writeln('Khong tim thay tep OverTest.Ovr');
         halt;
   end;
end.
```

• Tệp chương trình chính OverTest.pas:

```
{$O+,F+}
uses overlay, OvrStart, overl1, overl2, crt;
{$O Overl1}
{$O Overl2}
begin  clrscr;
       ThuTuc1;
       ThuTuc2; readln;
end.
```

d) Các hàm và thủ tục mẫu trong Unit Overlay

Hệ thống overlay của Turbo Pascal có 5 chương trình con để khởi động trình quản lý overlay và điều khiển cách dùng bộ nhớ của chương trình. Khi thực hiện chúng, các chương trình con này ấn định biến toàn cục **OvrResult** để cho biết có xảy ra vấn đề gì không. Biến này có thể nhận các hằng số sau:

```
Const  OvrOK = 0; không có lỗi
       OvrError = -1 ; lỗi không xác định
       OvrNotFound = -2 ; không tìm thấy tệp overlay
       OvrNotMemory = -3 ; không đủ bộ nhớ
       OvrIOError = -4 ; lỗi đọc tệp .OVR
       OvrNoEMSDriver = -5 ; chưa nạp trình điều khiển EMS
       OvrNoEMSMemory = -6 ; không đủ bộ nhớ EMS.
```

Mỗi lần dùng các chương trình con của unit Overlay nên kiểm tra giá trị của biến OvrResult để biết có lỗi gì không.

• **Thủ tục OvrInit**(FileName: string). Thủ tục này khởi động trình quản lý overlay và chuẩn bị các unit bị overlay cần dùng. FileName là tên tệp overlay, thường là tên của tệp chương trình chính với đuôi OVR. Thủ tục này được gọi duy nhất một lần và phải được gọi trước bất kỳ unit bị overlay nào.

• **Thủ tục OvrInitEMS** nạp tệp overlay vào vùng nhớ mở rộng nếu đủ vùng nhớ mở rộng khả dụng (giúp cho tốc độ chạy chương trình nhanh hơn). Nếu không đủ, máy vẫn đọc các unit bị overlay trên đĩa.

• **Thủ tục OvrSetBuf**(BufSize: longint) cấp phát BufSize byte cho vùng đệm overlay. Khi dùng thủ tục OvrInit(), trình quản lý overlay cần lượng vùng nhớ nhỏ nhất để thực hiện các thủ tục bị overlay, thủ tục OvrSetBuf() nhằm mở rộng vùng này. Ví dụ

lệnh OvrSetBuf(100000) để dành 100.000 byte cho vùng đệm overlay. BufSize phải lớn hơn số byte của vùng đệm tối thiểu do máy bố trí và nhỏ hơn MemAvail. Thủ tục OvrSetBuf chỉ được gọi một lần duy nhất khi xoá vùng Heap của các biến cấp phát động.

• **Hàm OvrGetBu**.. longint. Hàm trả về kích thước của vùng đệm overlay. Ta có thể dùng giá trị này làm chuẩn khi cần tăng kích thước vùng đệm bằng OvrSetBuf().

• **Thủ tục OvrClearBuf** nhằm xoá tất cả các unit bị overlay. Điều này đảm bảo bất kỳ lời gọi kế tiếp nào đến một thủ tục bị overlay đều yêu cầu đọc đĩa. Thông thường không bao giờ cần xoá vùng đệm overlay trừ khi ta muốn dùng vùng nhớ này vào mục đích khác.

Bài tập

1. Lập một unit ThuVien1.tpu gồm hàm tìm ước số chung lớn nhất, tìm bội số chung nhỏ nhất của hai số nguyên, và thủ tục sắp xếp một mảng số thực theo thứ tự tăng hoặc giảm dần. Tiếp theo dịch tệp ThuVien1.pas ra tệp ThuVien1.tpu, ta được Unit ThuVien1. Dùng unit này giải ba bài toán sau:

a) Cho số nguyên n. Tìm tất cả các phân số tối giản dạng i / j với 1<= i, j <= n.

Thuật toán : với mỗi phân số i / j , nếu ước số chung lớn nhất của i và j là 1 thì phân số là tối giản.

b) Nhập vào từ bàn phím một mảng số thực gồm n phần tử, sắp xếp mảng theo thứ tự giảm dần.

c) Tìm USCLN và BSCNN của n số nhập vào từ bàn phím.

2. Lập một unit ThuVien2.tpu có các thủ tục và hàm sau:

- Thủ tục Tong(n, A, B, C): tính mảng C qua các mảng A và B theo công thức C[i] = A[i] + B[i], i = 1, . . ., n.
- Thủ tục Hieu(n, A, B, D): tính mảng D qua các mảng A và B theo công thức D[i] = A[i] - B[i], i = 1, . . ., n.
- Hàm vô hướng VoHuong(n, A, B) tính tích vô hướng của hai véc tơ n chiều A và B.
- Hàm tính giai thừa của một số nguyên dương k GiaiThua(k)

Dùng Unit ThuVien2.tpu để giải hai bài toán:

a) Nhập từ bàn phím hai véc tơ A và B đều có n phần tử, n cũng nhập từ bàn phím. Tính các véc tơ C và D, in hai véc tơ này. Tính và in ra tích vô hướng của hai véc tơ A và B.

b) Tính S = 1 + 1 / 2! + 1 / 3! + ... + 1 / n!

3. Xây dựng một unit gồm các hàm tính: diện tích hình tam giác, diện tích hình thang, chu vi đường tròn, diện tích hình tròn, diện tích mặt cầu, diện tích xung quanh khối tứ diện đều, thể tích khối tứ diện đều . . . Dịch unit này ra tệp có đuôi TPU. Dùng unit để tính chu vi, diện tích, thể tích các hình trên với số liệu nhập vào từ bàn phím.

Chương 11

Unit DOS

1. Thiết lập các tham số của hệ điều hành DOS

• **Hàm DosVersion**: word trả về một word biểu thị version của DOS, byte thấp chứa phần nguyên, byte cao chứa phần lẻ sau dấu chấm.

Chương trình **Version.pas** nhằm xem version của DOS:

```
uses DOS;
var Ver: Word;
begin Ver := DosVersion;
 Writeln('This is DOS version ', Lo(Ver), '.',Hi(Ver)); readln;
end.
```

• **Thủ tục SetCBreak**(Break: boolean): đặt trạng thái của việc kiểm tra tổ hợp phím Ctl+C trong DOS. Nếu trạng thái là False, DOS chỉ kiểm tra Ctrl+C trong khi vào hay ra đối với màn hình, máy in và các thiết bị nối tiếp. Khi trạng thái là True, DOS kiểm tra Ctrl+C để dừng chương trình mỗi khi có lời gọi tới hệ thống.

• **Thủ tục GetCBreak**(Break: boolean): trả về tình trạng của việc kiểm tra Ctrl+C trong DOS.

Chương trình **CtrlC1.pas** : thay đổi trạng thái của việc kiểm tra Ctrl+C.

```
uses crt,Dos;
const OffOn : array [Boolean] of String[3]=('OFF','ON');
var  cb : Boolean;
begin  clrscr; delay(1000);  GetCBreak(cb);
  Writeln('Kiem tra Ctrl+C la: ',OffOn[cb]);  cb := not(cb);
  Writeln('Trang thai moi cua kiem tra Ctrl+C la: ',OffOn[cb]);
  SetCBreak(cb); readln;
end.
```

Chương trình **CtrlC2.pas** : chương trình có thể dừng khi ấn Ctrl+C.

```
uses crt,dos;
var i,j: integer; OldBreak: boolean;
begin clrscr; GetCBreak(OldBreak); writeln(OldBreak);
CheckBreak:= False; SetCBreak(True);
for i:=1 to 1000 do
  begin write(i,'  '); delay(10); j:= DosVersion; end;
SetCBreak(OldBreak);
end.
```

Giải thích. Trong Unit CRT có biến CheckBreak = True / False nhằm cho phép dùng tổ hợp phím Ctrl+Break để ngắt hay không ngắt các lệnh Write và Writeln khi chương trình đang chạy (chương trình đã dịch ra đuôi EXE). Chương trình trên ngừng chạy nếu ta ấn Ctrl+C vì có sự kiểm tra Ctrl+C khi có lời gọi hàm DosVersion. Nếu trong dòng thứ 4 SetCBreak(True) thay là SetCBreak(False) thì ta không thể dừng chương trình bằng Ctrl+C được.

Môi trường của DOS là một vùng gồm các biến ký tự ASCII có dạng Var = Value. Các biến môi trường chuẩn như Comspec chỉ định đường dẫn tới tệp Command.com, biến Prompt thiết lập dấu nhắc của DOS, các biến môi trường khác ta có thể lập bằng lệnh SET. Ví dụ để tạo biến môi trường có tên TMP và gán tên ổ đĩa trong RAM là E cho nó thì ta dùng lệnh Set TMP = E:\ trực tiếp sau dấu nhắc của DOS hay trong tệp Autoexec.bat.

• **Hàm EnvCount**: integer trả về số lượng xâu chứa trong môi trường DOS. Mỗi xâu môi trường có dạng Var = Value. Có thể kiểm tra các xâu với hàm EnvStr.

• **Hàm EnvStr**(Index: integer) : string trả về một xâu môi trường nhất định từ môi trường DOS. Một xâu do EnvStr trả về có dạng Var = Value. Chỉ số của xâu đầu tiên là 1. Nếu chỉ số không nằm trong [1..EnvCount] thì hàm trả về xâu rỗng.

• **Hàm GetEnv**(EnvVar: string): string trả về giá trị của một biến môi trường nhất định. Tên biến có thể viết chữ hoa hay chữ thường nhưng không được có dấu bằng (=). Nếu biến môi trường không có thì hàm trả về xâu rỗng.

Chương trình **BienHT.pas**: xem các biến hệ thống.

```
uses crt,Dos;
var i,n: Integer;
begin   clrscr; n:= EnvCount;  writeln('So bien he thong = ',n);
for i := 1 to n do
   begin  Writeln(EnvStr(i)); if i mod 10 = 0 then readln; end;
writeln; writeln('Bien Comspec = ',GetEnv('COMSPEC'));
writeln('Bien Path = ',GetEnv('PATH'));
writeln('Bien Prompt = ',GetEnv('prompt'));
readln; end.
```

2. Làm việc với ngày tháng và thời gian

• **Thủ tục GetDate**(var Year, Month, Day, DayOfWeek: word) trả về một tập xác định ngày hiện tại lưu trong hệ thống. Phạm vi của các giá trị trả về là Year 1980..2099, Month 1..12, Day 1..31, DayOfWeek 0..6 (giá trị 0 tương ứng với chủ nhật).

• **Thủ tục SetDate**(Year, Month, Day: word) đặt lại ngày hiện tại trong hệ thống. Phạm vi giá trị của các biến như trong thủ tục GetDate.

Chương trình **Ngay.pas**: lấy ngày trong hệ thống và thay đổi.

```
uses Dos,crt;
const  days : array [0..6] of String[9] =
    ('Chu nhat','Thu hai','Thu ba','Thu tu','Thu nam','Thu sau','Thu bay');
var  y, m, d, dow, CK : Word;
begin clrscr;  GetDate(y,m,d,dow);
Writeln('Hom nay la ',days[dow],', ',d:0,'/',m:0,'/',y:0);
write('Co dat lai ngay khong (1/0) ? '); readln(CK);
if CK=1 then
  begin write('- Vao ngay : '); readln(d);
        write('- Vao thang : '); readln(m);
        write('- Vao nam : '); readln(y);  SetDate(y,m,d);
  end;
end.
```

• **Thủ tục GetTime**(Hour, Minute, Second, Sec100: word) trả về tập hợp xác định thời gian hiện tại lưu trong hệ thống. Phạm vi các giá trị trả về là Hour 0..23, Minute 0..59, Sec100 (phần trăm của giây) 0..99.

• **Thủ tục đặt SetTime**(Hour, Minute, Second, Sec100: word) đặt giờ hiện tại cho hệ thống. Phạm vi hợp lệ của các tham số như trong thủ tục GetTime.

Chương trình **ThoiGian.pas** lấy thời gian trong hệ thống và thay đổi.

```
uses Dos;
var ck, h, m, s, hund: Word;
function LeadingZero(w: Word): String;
var  s : String;
begin   Str(w:0,s); if Length(s)= 1 then  s:= '0'+s;
  LeadingZero := s;
end;

begin  GetTime(h,m,s,hund);
Writeln('Bay gio la ',LeadingZero(h),':',
LeadingZero(m),':',LeadingZero(s),'.',LeadingZero(hund));
write('Co sua thoi gian khong 1/0 ? '); readln(ck);
if ck=1 then
  begin write('- Vao gio : '); readln(h);
        write('- Vao phut : '); readln(m);
        write('- Vao giay : '); readln(s); SetTime(h,m,s,0);
  end;
end.
```

Chương trình **RunTime.pas**: tính thời gian chạy một chương trình.

```
uses DOS;
var i,h,m,s,ms: word; t: real;
begin GetTime(h,m,s,ms); t:= (h*60 + m)* 60 + s + ms/100;
for i:=1 to 10000 do write(random:13:5,'  '); readln;
GetTime(h,m,s,ms); t:= (h*60 + m)* 60 + s + ms/100 - t;
ms:= Trunc(100 * Frac(t));  h:= Trunc(t) div 3600;
m:= Trunc(t-3600*h) div 60; s:= Trunc(t-3600*h) mod 60;
writeln('Thoi gian tinh toan la : ',
      h:1,' gio ',m:2,' phut ',s:2,'.',ms:2,' giay');
readln; end.
```

Hệ điều hành lưu trữ ngày và giờ mà tệp có sửa đổi mới nhất. Unit DOS cho phép xem và thay đổi thông tin này. Để tiết kiệm đĩa, DOS mã hoá ngày giờ thành hai giá trị 16 bit, về sau unit DOS dồn lại thành giá trị longint có 32 bit (dạng nén). Unit DOS định nghĩa kiểu dữ liệu DateTime như sau

DateTime = Record Year, Month, Day, Hour, Min, Sec : word ; End;
Phạm vi giá trị của các trường như trong hai thủ tục GetDate và GetTime.

• **Thủ tục GetFTime**(var F; var Time: longint) trả về ngày tháng ghi lần cuối cùng của tệp. F là một biến tệp đã được gán và mở. Thời gian được trả về trong Time cần phải giải nén bằng UnpackTime.

• **Thủ tục UnPackTime**(Time: longint; var T: DateTime) chuyển đổi thời gian ở dạng nén 32 bit chứa trong Time sang thời gian ở dạng bản ghi DateTime.

• **Thủ tục PackTime**(var T: DateTime; var Time: longint) nén thời gian ở dạng bản ghi DateTime sang dạng Longint.

• **Thủ tục SetFTime**(var F; Time: longint) đặt lại ngày và giờ (đã nén) cho tệp. F cần phải được gán và đã mở.

Chương trình **FileTime.pas** : xem thời gian tạo tệp và đặt lại.

```
uses dos;
var ck: integer; ftime: LongInt; dt,newDT: DateTime;
    f: file; fname: string[30];
begin write('Vao ten tep : '); readln(fname); assign(f,fname);
{$I-} Reset(f); {$I+}
if IOResult <> 0 then Halt; GetFTime(f,ftime); UnPackTime(ftime,dt);
with dt do
  begin writeln(day:1,'-',month,'-',year:1);
        writeln(hour:1,':',min,':',sec:1);
  end;
write('Co dat lai thoi gian cho tep khong 1/0 ? '); readln(ck);
if ck=1 then
begin
with newDT do
 begin write('- Ngay : '); readln(day);
   write('- Thang : '); readln(month);
   write('- Nam : '); readln(year); write('- Gio : '); readln(hour);
   write('- Phut : '); readln(min); write('- Giay : '); readln(sec);
 end;
PackTime(newDT,ftime); SetFTime(f,ftime); close(f);
end;
end.
```

3. Xem khả năng của đĩa

• **Hàm DiskFree**(Drive: byte): longint trả về số byte còn trống trên ổ đĩa chỉ định. Drive = 0 để chỉ ổ đĩa đang làm việc. 1 là ổ đĩa A. 2 là ổ đĩa B. 3 là ổ C. Giá trị trả về là -1 nếu số chỉ ổ đĩa là không hợp lệ.

• **Hàm DiskSize**(Drive: byte): longint trả về cỡ của đĩa tính theo byte. Các giá trị của Drive như trong hàm DiskFree. Nếu số của đĩa không hợp lệ hàm cũng trả về giá trị -1.

Chương trình **Disk.pas**: xem cỡ đĩa, phần còn trống, phần đã sử dụng.

```
uses Dos;
var s1,s2: longint; d: byte;
begin  writeln('Quy uoc: 0 - o dia ngam dinh,'+
               ' 1 - o A, 2 - o B, 3 - o C');
write('Vao ten o dia: '); readln(d);
s1:= DiskSize(d) div 1024; s2:= DiskFree(d) div 1024;
Writeln('Do lon dia : ',s1,' Kbyte');
Writeln('Con trong : ',s2, ' Kbyte');
Writeln('Da dung : ',s1-s2, ' Kbyte');
readln; end.
```

4. Làm việc với thư mục và tệp

• **Biến DosError**: integer dùng trong nhiều thường trình của unit DOS để chứa các lỗi. Các mã của biến DosError: 0 - không có lỗi, 2 - không tìm thấy tệp, 3 - không tìm thấy đường dẫn, 5 - không thể truy nhập được, 6 - con trỏ tệp không hợp lệ, 8 - không đủ bộ nhớ, 10 - môi trường không hợp lệ, 11 - dạng thức không hợp lệ, 18 - không còn tệp nào nữa.

• **Các hằng số thuộc tính tệp**. Thuộc tính tệp lưu trong một byte, tuỳ theo các bit từ 0 đến 5 nhận giá trị 0 hay 1 mà ta biết được thuộc tính của tệp. Bảng dưới cho các hằng số về thuộc tính tệp:

```
CONST
    ReadOnly    = $01 ;    bit 0, tệp chỉ để đọc
    Hidden      = $02 ;    bit 1, tệp ẩn
    SysFile     = $03 ;    bit 2, tệp hệ thống
    VolumID     = $08 ;    bit 3, nhãn đĩa
    Directory   = $10 ;    bit 4, thư mục con
    Archive     = $20 ;    bit 5, thuộc tính lưu trữ
    AnyFile     = $3F ;    bit 0-5, tổng tất cả các thuộc tính trên
```

Ta có thể cộng các thuộc tính trên để được thuộc tính mong muốn cho tệp.

• **Kiểu dữ liệu DirStr, NameStr, ExtStr** được định nghĩa:

```
TYPE  ComStr = string[127];  cho một dòng lệnh
      PathStr = string[79];  tên tệp đầy đủ
      DirStr = string[67];   cho đường dẫn trên đĩa
      NameStr = string[8];   cất giữ tên tệp
      ExtStr = string[3];    cất giữ phần mở rộng tên tệp
```

• **Kiểu dữ liệu SearchRec** dùng trong các thủ tục FindFirst và FindNext:

```
TYPE  SearchRec = Record
                    Fill : aray[1..21] of byte ;  trường hệ thống
                    Attr : byte ;                 trường thuộc tính
                    Time : Longint;               ghi thời gian ở dạng nén
                    Size : longint;               độ lớn của tệp
                    Name : string[12];            tên tệp
                  End;
```

• **Thủ tục FindFirst**(Path: string; Attr: word; var S: SearchRec): tìm trong thư mục đã chỉ định Path hoặc trong thư mục hiện hành tệp đầu tiên phù hợp với tên tệp và tập các thuộc tính đã chỉ định trong Attr. Các hằng số Attr và kiểu SearchRec đã cho ở trên. Nếu tìm thấy DosError cho giá trị 0.

• **Thủ tục FindNext**(var S: SearchRec): tìm tệp tiếp theo phù hợp với tên và thuộc tính đã chỉ định trong lời gọi FindFirst trước. Nếu tìm thấy hàm DosError cho giá trị 0.

Khi dịch chương trình **LietKe.pas** sau ra tệp LietKe.exe và chạy, nó sẽ liệt kê các tệp *.pas trong thư mục `C:\tp7\bin`.

```
uses crt,Dos;
```

```
var DirInfo: SearchRec; i: integer;
begin clrscr; i:=1; delay(1000);
FindFirst('c:\tp7\bin\*.pas', Archive, DirInfo);
while DosError = 0 do
  begin Writeln(DirInfo.Name,'  ',DirInfo.size); inc(i);
        if i mod 10 = 0 then readln; FindNext(DirInfo);
  end;
readln; end.
```

• **Hàm FExpand**(Path: PathStr): PathStr mở rộng tên tệp (hoặc một phần đường dẫn và tên tệp) trong Path thành tên đầy đủ viết hoa bao gồm: tên ổ đĩa, gạch chéo ngược, đường dẫn tới thư mục chứa tệp, tên tệp. Giả sử thư mục hiện hành là C:\TEST, bảng sau giải thích cách dùng hàm:

Tham số Path	Giá trị trả về
'test.txt'	'C:\TEST\TEST.TXT'
'\LEX\our.txt'	'C:\LEX\OUR.TXT'
'\'	'C:\'
'SUBDIR'	'C:\TEST\SUBDIR\'
'A:\LEX\our.txt'	'A:\LEX\OUR.TXT'

• **Thủ tục FSplit**(F: PathStr; var Dir: DirStr; var Name: NameStr; var Ext: ExtStr): tách đường dẫn cùng tên tệp trong F thành 3 phần: đường dẫn, phần tên chính của tệp, phần mở rộng. Bảng sau giải thích cách dùng thủ tục:

Tham số F	Dir	Name	Ext
'C:\DOS\SYS.COM	'C:\DOS\'	'SYS'	'.COM'
'BILL.EXE'	''	'BILL'	'.EXE'
'GAMES\DIGGER'	'GAMES\'	'DIGGER'	''
'GAME\DIGGER\'	'GAME\DIGGER\'	''	''
'A:FILE.COM'	'A:'	'FILE'	'.COM'

Chương trình **FSplit.pas** liệt kê tất cả các tệp trong thư mục hiện hành cùng với thời gian tạo tệp.

```
uses dos,crt;
var s: string; DS: DirStr; FN: NameStr; EN: extStr;
    SR: SearchRec; DT: DateTime;

function PadRight(s: string; L: word): string;
begin if Length(s) > L then s[0]:= chr(L);
  while Length(s) < L do s:=s+' ';   PadRight:= s;
end;

BEGIN clrscr;  FindFirst('*.*',AnyFile,SR);
while DosError=0 do
begin  with SR do
         begin UnPackTime(Time,DT);
            with DT do
              begin
                s:= FExpand(Name);FSplit(S,DS,FN,EN);
                writeln(PadRight(DS,15),' ',PadRight(FN,9),' ',
                PadRight(EN,5),' ',Size:7,' ',Day:2,'-',Month:2,
                '-',Year:4,' ',Hour:2,':',Min:2,':',Sec:2);
              end;
         end;
       FindNext(SR);
```

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

• **Hàm FSearch**(Path: PathStr: DirList: string) : PathStr tìm kiếm một tệp chỉ định bởi Path trong danh sách các thư mục DirList. Các thư mục trong DirList cần phải tách nhau bởi dấu chấm phẩy. Lời gọi hàm FSearch có thể có dạng sau:

```
S:= FSearch('MyFile.doc','C:\;C:\VANBAN;A:\');
S:= FSearch('VIDU.PAS',''); tìm trong thư mục hiện hành.
```

Chương trình **FSearch.pas** : tìm tệp Turbo.exe trong các thư mục liệt kê ở lệnh PATH (thuộc tệp Autoexec.bat) và trả về đường dẫn đầy đủ.

```
uses Dos;
var  S: PathStr;
begin  S := FSearch('TURBO.EXE',GetEnv('PATH'));
if S = '' then  WriteLn('TURBO.EXE not found')
          else  WriteLn('Found as ',FExpand(S));
readln;end.
```

• **Thủ tục GetFAttr**(var F; var Attr: word) cho thuộc tính của tệp. F là biến tệp (có kiểu, không kiểu hay tệp văn bản) mà đã được gán nhưng chưa mở.

Xác định thuộc tính của tệp đưa vào theo tham số dòng lệnh. Giả sử tệp chương trình là **LayTT.pas**, đã dịch ra tệp LayTT.exe, muốn xem thuộc tính của tệp Vidu.pas ta dùng lệnh: LayTT Vidu.pas ↵

```
uses Dos;
var F: file;  Attr: Word;
begin  { Lấy tên tệp từ dòng lệnh }
Assign(F, ParamStr(1)); GetFAttr(F, Attr); Writeln(ParamStr(1));
if DosError <> 0 then  Writeln('DOS error code = ', DosError) else
begin Write('Attribute = ', Attr);
 if Attr and ReadOnly <> 0 then Writeln('Read only file');
 if Attr and Hidden <> 0 then Writeln('Hidden file');
 if Attr and SysFile <> 0 then Writeln('System file');
 if Attr and VolumeID <> 0 then Writeln('Volume ID');
 if Attr and Directory <> 0 then Writeln('Directory name');
 if Attr and Archive <> 0 then Writeln('Archive (normal file)');
end;
end.
```

Giải thích. Hàm *ParamStr(Index: word) : string* trả về tham số thứ Index từ dòng lệnh, hoặc trả về một xâu rỗng nếu Index lớn hơn ParaCount (hàm cho số lượng tham số trong dòng lệnh cần chuyển cho chương trình). ParamStr(0) trả về đường dẫn và tên tệp của chương trình đang chạy (ví dụ `C:\BP\MyProg.exe`).

• **Thủ tục SetFAttr**(var F; Attr: word) đặt thuộc tính cho tệp. Lỗi thông báo trong biến DosError. Các mã lỗi cho phép là 3 và 5, tệp F cần không mở.

Chương trình **DatTt.pas**: đặt thuộc tính cho tệp mà tên gõ vào từ bàn phím.

```
uses Dos;
var  F: file; tf: string[30];
begin  write('Vao ten tep : '); readln(tf); Assign(F,tf);
SetFAttr(F, ReadOnly+ Hidden+ Archive); Readln;
end.
```

5. Sử dụng ngắt trong Turbo Pascal

a) Các thanh ghi của 8086

Các bộ vi xử lý ngày nay so với thế hệ ban đầu đã phát triển rất nhiều, nhưng về nguyên tắc vẫn dựa trên cơ sở của bộ vi xử lý 8086. Các thanh ghi trong CPU của 8086 có chiều dài 16 bit (đánh số từ 0, 1, . . ., 15) và được gán tên. Tuỳ theo công dụng người ta chia thành các nhóm sau:

- Các thanh ghi đa dụng AX (accumulator), BX (base), CX (count), DX (data) dùng để chứa kết quả hiện thời. Chúng có thể dùng như một thanh ghi 16 bit hoặc hai thanh ghi 8 bit. Ví dụ AX là 16 bit, AL là 8 bit thấp của AX, AH là 8 bit cao của AX. Trừ 4 thanh ghi đa dụng, các thanh ghi còn lại đều dùng 16 bit.

- Các thanh ghi đoạn CS (code segment), DS (data segment), SS (stack segment), ES (extra segment) lưu địa chỉ đoạn.

- Các thanh ghi offset IP (instruction pointer), SP (stack pointer), BP (base pointer), SI (source index), DI (destination index): dùng kết hợp với các thanh ghi đoạn để định vị bộ nhớ. Thanh ghi IP chứa địa chỉ lệnh đang thực hiện .

- Thanh ghi cờ FLAGS: chứa 9 cờ (mỗi cờ chiếm 1 bit), dùng để xác định trạng thái của bộ vi xử lý. Các cờ liên quan tới Turbo Pascal: cờ nhớ CF (carry flag, bit 0) thiết lập (bằng 1) khi có nhớ trong phép tính, cờ chẵn lẻ PF (parity, bit 2) thiết lập khi số là chẵn, cờ phụ AF (auxiliary, bit 4), cờ zero ZF (zero, bit 6) thiết lập khi kết quả tính bằng 0, cờ dấu SF (sign, bit 7) thiết lập khi kết quả tính là âm, cờ tràn OF (overflow, bit 11) thiết lập khi bị tràn số.

Trong Unit DOS định nghĩa sẵn các kiểu dữ liệu và hằng số liên quan tới các thanh ghi.

• Kiểu Registers dùng để truy nhập các thanh ghi, dùng trong các thủ tục INTR, MsDos.

```
Type  Registers = Record
                   case integer of
                      0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: word);
                      1: (AL,AH,BL,BH,CL,CH,DL,DH: byte);
                End;
```

• Các hằng cờ hiệu:

```
CONST  Carry = $0001;     FParity = $0004;  FAuxiliary = $0010;
       FZero = $0040;     FSign = $0080;    FOverflow = $0800;
```

b) Các khái niệm về ngắt

Ngắt là khả năng dừng chương trình chính đang chạy để thực hiện một chương trình con khác gọi là chương trình con ngắt. Các chương trình con ngắt của hệ thống được ROM BIOS hoặc DOS cài đặt sẵn trong bộ nhớ mà địa chỉ Segment và Offset của từng chương trình được quản lý bằng cách ghi vào vùng nhớ 4 byte (địa chỉ offset đặt trong 2 byte thấp, địa chỉ segment đặt trong 2 byte cao) gọi là véc tơ ngắt. Mỗi véc tơ ngắt là một con trỏ xa (far pointer) 4 byte trỏ đến đoạn chương trình ngắt.

Các vector ngắt chứa địa chỉ của các chương trình ngắt được bố trí liên tiếp nhau trong bộ nhớ tạo thành một bảng gọi là bảng vector ngắt. Bảng vector ngắt có 256

vector thành phần được đánh số theo thứ tự từ 0 đến 255 (hay $00 đến $FF) có độ lớn 256 x 4 byte = 1024 byte và được đặt trong bộ nhớ bắt đầu từ địa chỉ 0000:0000h đến 0000:03FFh (phần thấp nhất của địa chỉ vật lý). Số thứ tự của ngắt được gọi tương ứng với số thứ tự của các phần tử trong Bảng vector ngắt, do đó vị trí của ngắt số 0 đặt ở phần tử số 0 của Bảng vector ngắt, tức là các ô nhớ từ 0 đến 3h. Như vậy 4 byte tiếp theo chứa địa chỉ của ngắt 1, . . ., địa chỉ của ngắt cuối cùng 255 chứa trong các ô nhớ từ 3FCh đến 3FFh. Khi biết một số hiệu ngắt, để tính vị trí ô nhớ mà tại đó lưu trữ vị trí bắt đầu của một ngắt ta chỉ cần lấy số hiệu ngắt đó nhân với 4.

Có hai loại ngắt: ngắt cứng và ngắt mềm. Mặc dù có cấu trúc tương tự và cùng chia nhau bảng vector ngắt, chúng có các khác nhau cơ bản.

Ngắt mềm là một ngắt từ trong chương trình, nó được gọi bằng một chỉ thị INT trong ngôn ngữ Assembler hay thủ tục INTR của Turbo Pascal. Có thể xem ngắt mềm như một chương trình con.

Các ngắt cứng được kích hoạt không phải từ trong chương trình mà do từ các thiết bị phần cứng của máy tính. Có hai loại ngắt cứng:

- Ngắt cứng bên trong: ngắt này tác động khi trong chương trình gặp một sự kiện nào đó mà máy tính không thể thực hiện được như chia cho một số không (ngắt số 0).

- Ngắt cứng bên ngoài: ngắt này tác động do các thiết bị ngoại vi của máy tính. Ví dụ khi ta bấm một phím thì bàn phím gây nên một ngắt. Các ngắt này định hướng theo sự kiện: phím bị ấn, việc đọc đĩa kết thúc, máy in in xong một ký tự . . . Sau khi chương trình ngắt kết thúc, chương trình chính được thực hiện tiếp.

c) Các thủ tục Intr và MsDos để gọi ngắt

Một chương trình ngắt có thể có nhiều chức năng riêng, mỗi chức năng được đặt tên là một số nguyên (gọi là số hiệu ngắt). Một chức năng của ngắt khi thực hiện sẽ yêu cầu nhận một số dữ liệu từ các thanh ghi, sau khi thực hiện xong ngắt kết quả cũng được lấy ra từ các thanh ghi. Như vậy một chức năng của ngắt chính là một chương trình con nhưng nó không có tham số như trong Pascal. Nhiệm vụ của từng chức năng, dữ liệu yêu cầu cung cấp và dữ liệu trả về được hướng dẫn cụ thể trong các tài liệu chỉ dẫn kỹ thuật của máy tính.

Unit DOS cung cấp hai thủ tục để gọi ngắt:

• **Thủ tục Intr**(IntNo: byte; var Regs: Registers) thực hiện ngắt mềm IntNo (có giá trị từ 0 đến 255). Dữ liệu đưa vào và kết quả lấy ra đều ở trong biến Regs có kiểu Registers.

• **Thủ tục MsDos**(var Regs: Registers) gọi ngắt thứ $21 (gọi các chương trình của hệ điều hành DOS). Lời gọi MsDos(Regs) tương đương với lời gọi Intr($21,Regs).

Để gọi ngắt chúng ta thực hiện các bước sau:

- Khai báo biến Regs kiểu Registers;

- Thiết lập giá trị các thanh ghi thông qua các trường của biến Regs: gán số hiệu chức năng cho trường AH của biến Regs, gán số hiệu phụ nếu có cho trường AL của biến Regs, gán các dữ liệu theo yêu cầu của chức năng vào các trường của biến Regs theo quy định;

- Gọi ngắt bằng thủ tục Intr(IntNo, Regs) hay MsDos(Regs);

- Nhận kết quả từ các trường của Regs.

Chương trình **Intr.pas** minh hoạ cách dùng thủ tục Intr để lấy ngày hệ thống, sử dụng chức năng $2A của ngắt $21. Dữ liệu vào: AH = $2A. Kết quả lấy ra: CX chứa năm, DH chứa tháng, DL chứa ngày.

```
uses Crt,Dos;
var date, year, month, day: string; regs: Registers;
begin clrscr; regs.ah := $2a; with regs do intr($21,regs);
with regs do begin str(cx,year); str(dh,month); str(dl,day) end;
date := month+'/'+day+'/'+year;
writeln('Today''s date is ', date); readln;
end.
```

Chương trình **MsDos.pas** dùng thủ tục MsDos lấy ngày hệ thống.

```
uses crt,Dos;
var date, year, month, day: string; regs: Registers;
begin  clrscr; regs.ah := $2a; with regs do msdos(regs);
with regs do begin str(cx ,year); str(dh ,month); str(dl,day) end;
date := month+'/'+day+'/'+year;
writeln('Today''s date is ', date); readln;
end.
```

Chương trình **GetChar.pas**: đọc ký tự từ màn hình tại vị trí con trỏ. Dùng chức năng $08 của ngắt $10. Dữ liệu vào: đặt AH=$08, BH chứa trang màn hình. Kết quả: AL chứa mã ASCII của ký tự cần đọc. Chương trình sau in chữ Pascal theo hàng ngang, sau đó đọc từng ký tự và in chúng theo hàng dọc.

```
Uses Crt, Dos;
Var i: Byte; Ch: Char;
Function GetScrChar: Char;
Var  Regs: Registers;
Begin With Regs do
 Begin AH := $08; BH := 0; Intr($10,Regs);GetScrChar := Chr(AL); End;
End;

Begin ClrScr; Writeln('Pascal');
For i:=1 to Length('Pascal') do
  Begin GotoXY(i,1); Ch := GetScrChar; GotoXY(10,i); Write(Ch);
  End; Readln;
E
```

Chương trình **SetCurs.pas**: đặt lại kích thước con trỏ. Dùng chức năng $01 của ngắt $10. Dữ liệu vào: AH=$01, CH= Top (giá trị dòng quét trên) , CL= Bottom (giá trị dòng quét dưới), nếu CH=$20 (bit thứ 5 bằng 1) thì con trỏ biến mất. Con trỏ gồm các dòng nằm ngang được đánh số từ 0 đến 13.

```
Uses Crt, Dos;
Procedure SetCursor(Top,Bottom: Word);
Var  Regs: Registers;
Begin  Regs.AH := $01; Regs.CH := Top;
       Regs.CL := Bottom;  Intr($10,Regs);
End;

Begin Clrscr; Writeln('Giau con tro !'); SetCursor($20,$20); Readln;
Writeln('Con tro lon !'); SetCursor(0,13); Readln;
Writeln('Con tro nho !'); SetCursor(12,13); Readln;
End.
```

6. Thực hiện lệnh DOS từ trong Pascal

• **Dẫn hướng biên dịch { $M StackSize, HeapMin, HeapMax }** quy định các tham số phân chia bộ nhớ cho ứng dụng hay thư viện. StackSize là một số nguyên nằm giữa 1024 đến 65520 xác định kích thước của ngăn xếp (giá trị ngầm định là 16384 byte). HeapMin (HeapMax) xác định độ lớn cực tiểu (cực đại) của vùng Heap. HeapMin nằm trong khoảng từ 0 đến 655360, HeapMax nằm trong khoảng từ HeapMin đến 655360.

• **Hàm DosExitCode**: word cho mã trở về từ DOS khi thực hiện một công việc thuộc môi trường DOS trong lòng một chương trình Pascal. Mã 0 - chấm dứt bình thường, mã 1 - chấm dứt do ấn Ctrl+C, 2 - chấm dứt do lỗi thiết bị, 3 - chấm dứt do thủ tục Keep.

• **Thủ tục SwapVectors**: dùng trước và sau khi gọi thủ tục Exec nhằm bảo đảm sự toàn vẹn của bảng vector ngắt.

• **Thủ tục Exec**(Path, CmdLine: string) thực hiện chương trình ghi trong Path với tham số dòng lệnh là CmdLine. Lỗi trả về trong biến DosError. Khi dịch chương trình dùng Exec ta hãy giảm cỡ HeapMax, nếu không có thể xảy ra không đủ bộ nhớ. Nếu chương trình không dùng biến cấp phát động hãy giảm HeapMax xuống bằng 0.

Chương trình **Exec1.pas** cho phép chạy một lệnh DOS (kể cả chạy một chương trình có tham số dòng lệnh).

```
{$M $4000,0,0 }    { 16K stack, no heap }
uses Dos;
var  ProgramName, CmdLine: string;
begin Write('Program to Exec (full path): '); ReadLn(ProgramName);
Write('Command line to pass to ',ProgramName, ': '); ReadLn(CmdLine);
WriteLn('About to Exec...');
SwapVectors; Exec(ProgramName, CmdLine); SwapVectors;
WriteLn('...back from Exec');
if DosError <> 0 then WriteLn('Dos error #', DosError)
else
WriteLn('Exec successful.','Child process exit code = ',DosExitCode);
readln; end.
```

Chẳng hạn muốn thực hiện lệnh DOS

```
c:\>c:\pctex\ne.com c:\pctex\baocao.tex
```

ta cần nhập hai xâu ký tự: `'c:\pctex\ne.com'` và `'c:\pctex\baocao.tex'`

Muốn thực hiện lệnh DOS `c:\>copy bc1.txt+bc2.txt baocao.txt` ↵ ta cần nhập hai xâu ký tự `'c:\command.com'` và `'/c copy bc1.txt+bc2.txt baocao.txt'`

Chương trình **Exec2.pas** cho phép thực hiện chương trình Command.com. Chẳng hạn khi muốn thực hiện lệnh `dir *.prg` của DOS ta chỉ cần nhập vào xâu ký tự `'dir *.prg'`

```
{$M 8192,0,0}
uses Dos;
var Command: string[79];
begin Write('Enter DOS command: ');  Readln(Command);
if Command <> '' then Command := '/C ' + Command; SwapVectors;
Exec(GetEnv('COMSPEC'), Command); SwapVectors;
```

```
if DosError <> 0 then Writeln('Could not execute COMMAND.COM');
readln; end.
```

7. Lập trình thường trú

Chương trình thường trú (terminate and stay resident) là một chương trình sau khi cài đặt nó nằm thường trú trong bộ nhớ RAM, sau đó nó có thể được kích hoạt bằng một phím nóng. Một số chương trình điển hình loại này là Norton Commander, các chương trình canh phòng virus. Ta hãy xét một số công cụ để lập trình thường trú.

• **Cú pháp khai báo một thủ tục ngắt**:

```
Procedure IntProc(Flags, CS, IP, AX, BX, CX, DX, SI, DI, DS, ES, BP:
                  word); interrup;
```

Các thanh ghi xem như các giả tham số, ta có thể dùng và sửa đổi chúng trong thân của thủ tục ngắt. Thủ tục ngắt sẽ được thực hiện khi có một lời gọi ngắt tương ứng.

• **Thủ tục GetIntVec**(IntNo: byte; var Vector: pointer) trả về địa chỉ được lưu trong vector ngắt IntNo. Tham số IntNo chỉ số hiệu vector ngắt (0..255), và địa chỉ được trả về trong Vector.

• **Thủ tục SetIntVec**(IntNo: byte; Vector: pointer) đặt vector ngắt IntNo theo địa chỉ đã định Vector. IntNo là số hiệu vector ngắt (0..255), Vector chỉ địa chỉ. Sau lệnh này, mỗi khi gọi ngắt IntNo thì chương trình bắt đầu ở địa chỉ Vector sẽ chạy thay vì chạy chương trình có sẵn của hệ điều hành. Trước khi khai thủ tục ngắt cần có khai báo {$F+} và khi kết thúc thủ tục ngắt phải có khai báo {$F-}.

Vector thường được xây dựng với toán tử @ để sinh ra một địa chỉ của thủ tục ngắt. Giả sử Int1BSave là biến kiểu Pointer và Int1BHandler là tên của một thủ tục ngắt, dãy lệnh sau đây sẽ cài đặt ngắt $1B mới và sau đó khôi phục lại ngắt cũ:

```
GetIntVec($1B, Int1BSave);
SetIntVec($1B, @Int1BHandler); . . . . . .
SetIntVec($1B, Int1BSave);
```

• **Thủ tục Keep**(exitCode: Word) kết thúc chương trình và giữ nó thường trú trong bộ nhớ. ExitCode là trạng thái trở về, mã kết thúc của trạng thái trở về luôn bằng không.

• **Mảng Port và PortW** được định nghĩa sẵn trong Turbo Pascal để truy xuất các cổng dữ liệu. Cả hai đều là mảng một chiều, mỗi phần tử biểu thị một cổng dữ liệu mà địa chỉ của cổng tương ứng với chỉ số của nó. Kiểu của chỉ số là Integer hay Word. Các thành phần của mảng Port có kiểu byte, các thành phần của mảng PortW có kiểu Word.

• **Ba mảng Mem, MemL và MemW** được Turbo Pascal định nghĩa sẵn dùng để truy nhập trực tiếp bộ nhớ. Mỗi thành phần của Mem là byte, mỗi thành phần của MemW là Word, mỗi thành phần của MemL là Longint. Chỉ số của các mảng dùng cú pháp đặc biệt: dùng địa chỉ segment và offset cách nhau bởi dấu hai chấm. Ví dụ lấy byte ở địa chỉ B800:0000 ta dùng MEM[$B800:0000].

• **Ngắt bàn phím INT 9**. Khi một phím được ấn, ngắt 9 được gọi và sinh ra một mã quét (scancode). Mã quét được nhận từ cổng $60 của bàn phím. ROM BIOS đọc các mã quét và đổi thành trị ASCII, sau đó chuyển trị này vào vùng đệm bàn phím.

Việc đọc một phím thực chất là từ vùng đệm này. ROM BIOS còn duy trì một byte trạng thái bàn phím cho biết các phím chức năng có được ấn hay không. Byte trạng thái bàn phím được định vị ở địa chỉ 0000:0417h và gồm 8 bit, mỗi bit đóng vai trò như là một cờ hiệu báo cho hệ thống biết phím chức năng nào được ấn.

Chương trình thường trú **Resident.pas** làm nhiệm vụ: mỗi khi một phím được nhấn thì chương trình được kích hoạt và phát lên một âm thanh (nốt Sol với trường độ là 1/100 giây).

```
{$M $800,0,0 } { 2K stack, no heap }
{ This program causes a click each time a key is pressed.}
uses Crt, Dos;
var  KbdIntVec : Procedure;

{$F+}
procedure Keyclick; interrupt;
begin if Port[$60] < $80 then { Only click when key is pressed }
        begin Sound(392); Delay(10); Nosound;  end;
   inline ($9C); { PUSHF -- Push flags }
   { Call old ISR using saved vector } KbdIntVec;
end;
{$F-}

begin  { Insert ISR into keyboard chain } GetIntVec($9,@KbdIntVec);
   SetIntVec($9,Addr(Keyclick)); Keep(0);{ Terminate, stay resident }
end.
```

Chương trình thường trú **BaoGio.pas**: thông báo hết giờ làm việc trong môi trường DOS, chương trình cần dịch ra tệp BaoGio.exe. Giả sử ta muốn sau khi làm việc 15 phút (số này được đưa vào theo tham số dòng lệnh, giá trị ngầm định là 1 phút) thì trên dòng 1 màn hình xuất hiện dòng chữ 'Over time ! - Press F12 to continue' và chuông kêu liên tục cho đến khi ấn phím F12, khi đó ta chạy chương trình theo lệnh C:\>BAOGIO 15 ↵

```
{$M $800,0,0 }
Uses Crt,Dos;
Const  DefaultTime = 1; { Gio bao thuc mac dinh la 1 phut }
Var  CurrentTime : LongInt;    { Thoi gian hien tai }
     WorkTime : Word;    { Thoi gian can bao }
     KbdIntVec : Procedure;

{$F+}
Procedure KeyClick; Interrupt;
Begin Gotoxy(1,1); TextColor(Yellow); TextBackGround(Red);
Write(' Over time ! - Press F12 to continue. ');
{ Phat chuong neu phim F12 khong duoc bam }
While Port[$60] <> 88 do Write(#7);
SetIntVec($9,@KbdIntVec); { Phuc hoi ngat ban phim }
Inline($9C); KbdIntVec;
End;
{$F-}

{$F+}
Procedure TestTime; Interrupt;
```

```
Begin Inc(CurrentTime);
If ( CurrentTime >= Round(WorkTime*60*18.2) ) then
  Begin SetIntVec($1C,@KbdIntVec);{Phuc hoi vect ngat $1C cu}
     GetIntVec($9,@KbdIntVec); { Luu vect ngat ban phim }
     SetIntVec($9 ;KeyClick); { Xac dinh dia chi moi cho ngat ban phim }
  End;
Inline ($9C); KbdIntVec;
End;
{$F-}

Procedure TestParam;
Var WorkTimeStr: String; i: Byte; Code: Integer;
Begin  WorkTimeStr:='';
For i:=1 to ParamCount do WorkTimeStr:=WorkTimeStr + ParamStr(i);
Val(WorkTimeStr,WorkTime,Code);
{ Lay thoi gian mac dinh neu tham so khong hop le }
If Code <> 0 then WorkTime := DefaultTime;
End;

Begin TestParam;   { Xu ly tham so dong lenh }
CurrentTime:=0; GetIntVec($1C,@KbdIntVec); {Luu vect ngat $1C ban dau}
SetIntVec($1C,@TestTime); { Xac dinh dia chi moi cho vect ngat }
Keep(0);
End.
```

Giải thích:

- Thủ tục TestTime: xác định thời gian sau bao lâu cần thông báo hết giờ, thời gian này lấy từ tham số dòng lệnh và đưa vào biến Work Time. Nếu không có tham số dòng lệnh thì Work Time lấy bằng thời gian ngầm định là 1 phút.

- Chương trình sử dụng ngắt $1C (nhịp đồng hồ cho người sử dụng). Ngắt định thời $1C được gọi mỗi giây 18.2 lần bất chấp sự hoạt động của CPU. Chương trình báo giờ sẽ xây dựng trình sử lý ngắt mới cho ngắt $1C (thủ tục TestTime). Mỗi lần được gọi (sau 1/18.2 giây) thủ tục TestTime tăng biến Current Time lên 1 và so sánh với thời gian cần báo (tính theo khắc), nếu vượt quá thì nó thực hiện các công việc: phục hồi vector ngắt $1C cũ đã được lưu ở đầu chương trìhh, lưu vector ngắt bàn phím số 9, cho trình xử lý ngắt bàn phím trỏ đến thủ tục KeyClick.

- Thủ tục KeyClick sẽ in thông báo hết giờ ra màn hình và liên tục phát chuông cho đến khi phím F12 (có mã quét bàn phím là 88) được ấn thì khôi phục lại vector ngắt số 9.

Chương 12

Các tập hợp hữu hạn

1. Duyệt mọi hoán vị

Một tập hợp gồm n phần tử được đồng nhất với tập hợp {1, 2, . . ., n }, tức là ta cho tương ứng mỗi phần tử của tập hợp với một số tự nhiên. Ví dụ, tập hợp gồm 3 học sinh { Lan, Huệ, Cúc } đồng nhất với tập hợp { 1, 2, 3 }. Bài toán đặt ra là hãy duyệt tất cả các hoán vị của tập hợp gồm n phần tử { 1, 2, . . ., n }. Một hoán vị của n phần tử là một bộ gồm n phần tử được sắp xếp theo một trật tự nhất định, mỗi phần tử có mặt đúng một lần.

Phần tử thứ nhất của hoán vị có n cách chọn, sau khi chọn phần tử thứ nhất thì phần tử thứ hai có n-1 cách chọn, . . ., phần tử cuối cùng chỉ có 1 cách chọn. Do đó tổng số các hoán vị là n! hoán vị.

Để có thể duyệt hết các hoán vị, ta tìm cách sắp xếp các hoán vị theo quy tắc từ vựng. Ký hiệu một hoán vị là véc tơ P gồm n phần tử. Hoán vị P gọi là nhỏ hơn hoán vị Q nếu tồn tại chỉ số j sao cho P[i] = Q[i] với i = 1, . . ., j-1, P[j] < Q[j]. Sau đây là 24 hoán vị của tập hợp { 1, 2, 3, 4 }: (1 2 3 4), (1 2 4 3), (1 3 2 4), (1 3 4 2), (1 4 2 3), (1 4 3 2), (2 1 3 4), (2 1 4 3), (2 3 1 4), (2 3 4 1), (2 4 1 3), (2 4 3 1), (3 1 2 4), (3 1 4 2), (3 2 1 4), (3 2 4 1), (3 4 1 2), (3 4 2 1), (4 1 2 3), (4 1 3 2), (4 2 1 3), (4 2 3 1), (4 3 1 2), (4 3 2 1).

Thuật toán duyệt:

Bước 1. Xuất phát từ hoán vị nhỏ nhất P = (1, 2, 3, . . ., n). In hoán vị P ban đầu.

Bước 2. Theo hướng từ phần tử cuối cùng lùi về phần tử đầu tiên của hoán vị P, ta tìm chỉ số i đầu tiên thoả mãn P[i] < P[i+1] và chuyển sang bước 3. Nếu không tìm được chỉ số i như vậy thì thuật toán kết thúc, ta đã duyệt xong tất cả các hoán vị.

Bước 3. Tìm k là chỉ số sao cho P[k] = min {P[j] : j >= i+1, P[j] > P[i] }.

Bước 4. Đổi chỗ P[i] và P[k]. Đảo ngược dãy P[i+1], P[i+2], . . ., P[n] ta được hoán vị mới. Chuyển lên bước 2.

Chương trình **HoanVi1.pas** nhằm duyệt mọi hoán vị, khi chạy cần nhập vào số phần tử của tập hợp là n. Kết quả tính in ra màn hình và ra tệp HoanVi.kqu.

```
const nmax=20;
type mang=array[1..nmax] of integer;
var p:mang; d,t,i,k,j,n,min: integer; f: text;
begin
assign(f,'hoanvi.kqu'); rewrite(f); write('n = '); readln(n);
d:=1; for i:=1 to n do p[i]:=i;
write('Hoan vi ',d:3,' la: ');
for i:=1 to n do write(p[i]:3); writeln;
write(f,'Hoan vi ',d:3,' la: ');
for i:=1 to n do write(f,p[i]:3); writeln(f);
while true do
  begin i:=n-1;  while (i>1) and (p[i]>p[i+1]) do i:=i-1;
       if (i=1) and (p[1]>p[2]) then
```

```
                                begin close(f); readln; halt; end;
        min:=n+1;
        for j:=i+1 to n do
           if p[j]>p[i] then begin min:=p[j]; k:=j; end;
        t:=p[i]; p[i]:= p[k]; p[k]:=t;  k:=(n-i) div 2;
        for j:=1 to k do
           begin t:=p[i+j]; p[i+j]:=p[n-j+1]; p[n-j+1] := t; end;
        d:=d+1; write('Hoan vi ',d:3,' la: ');
        for i:=1 to n do write(p[i]:3); writeln;
        write(f,'Hoan vi ',d:3,' la: ');
        for i:=1 to n do write(f,p[i]:3); writeln(f);
        if d mod 23=0 then readln;
  end;
end.
```

2. Duyệt mọi tập con của một tập hợp

Cho một tập hợp gồm n phần tử $X = \{ x_1, x_2, \ldots, x_n \}$. Ta cho ứng mỗi tập con Y thuộc X với một bãy nhị phân độ dài n là $(b_1, b_2, \ldots, b_n)$, trong đó $b_i = 0$ nếu x_i không thuộc Y và $b_i = 1$ nếu x_i thuộc Y. Tổng số các dãy nhị phân có độ dài n là 2^n dãy.

Để duyệt được tất cả các tập con, ta tiến hành sắp xếp các dãy nhị phân theo thứ tự từ vựng giảm dần. Sau đây là tất cả các dãy nhị phân có 4 phần tử đã được sắp xếp theo thứ tự giảm dần: 1111, 1110, 1101, 1100, 1011, 1010, 1001,1000, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000. Dãy nhị phân 1010 có nghĩa là tập con bao gồm phần tử thứ nhất và phần tử thứ ba.

Thuật toán duyệt:

Bước 1. Xuất phát từ dãy nhị phân lớn nhất X = (1, 1, . . ., 1).

Bước 2. Theo hướng từ phần tử cuối cùng lùi về phần tử đầu tiên của dãy X, ta tìm chỉ số đầu tiên k sao cho X[k] = 1. Nếu không tìm được chỉ số k như thế, tức là lúc này X = (0, 0, . . ., 0), thì kết thúc thuật toán.

Bước 3. Thay X[k] = 0 và X[j] = 1 với j > k ta được dãy nhị phân mới. Quay lên bước 2.

Chương trình **TapCon1.pas**: duyệt mọi tập con. Khi chạy chương trình ta cần nhập vào số phần tử của tập hợp là n.

```
label 11;
const nmax=10;
type mang=array[1..nmax] of integer;
var x: mang; i,j,k,d,n: integer; f: text;
begin  assign(f,'tapcon.kqu'); rewrite(f);
write('n = '); readln(n);  d:=1;  for j:=1 to n do x[j]:=1;
11: write(f,'Tap thu ',d:4,' la: ');
for j:=1 to n do write(f,x[j]:3); writeln(f);
if d mod 23=0 then readln;  d:=d+1;
for i:=1 to n do
  begin if x[n-i+1]=1 then
          begin k:=n-i+1; x[k]:=0;
             if k<n then for j:=k+1 to n do x[j]:=1; goto 11; end;
  end;
close(f);  readln;
```

```
end.
```

3. Tìm tất cả các tập con k phần tử của tập gồm n phần tử

Cho một tập hợp A gồm n phần tử { 1, 2, 3, . . ., n } và một số k: 0 <= k <= n. Hãy duyệt tất cả các tập hợp con gồm k phần tử lấy từ tập A. Sau đây là danh sách tất cả các tập con gồm 4 phần tử lấy từ tập hợp { 1, 2, 3, 4, 5, 6 } đã được sắp xếp theo thứ tự từ vựng tăng dần: 1234, 1235, 1236, 1245, 1246, 1256, 1345, 1346, 1356, 1456, 2345, 2346, 2356, 2456, 3456

Mỗi một tập con k phần tử chính là một tổ hợp chập k của n phần tử. Số lượng tổ hợp là n! / [r! (n-r)!]. Chương trình **ToHop1.pas**: duyệt các tổ hợp.

```
const nmax=20;
type mang=array[1..nmax] of integer;
var a: mang; i,p,k,d,n: integer; f: text;

begin  assign(f,'tconk.kqu'); rewrite(f);
write('Tong so phan tu n    = '); readln(n);
write('So phan tu cua tap k = '); readln(k);
d:=0; for i:=1 to k do a[i]:=i; p:=k;
while p>=1 do
   begin d:=d+1; write(f,'Tap thu ',d:4,' la : ');
         for i:=1 to k do write(f,a[i]:3); writeln(f);
         if d mod 23=0 then readln; if a[k]=n then p:=p-1 else p:=k;
         if p>=1 then for i:=k downto p do a[i]:=a[p]+i-p+1;
   end;
close(f); readln;
end.
```

4. Duyệt mọi cách phân chia một tập hợp thành các tập hợp con

Cho một tập hợp gồm n phần tử { 1, 2, . . ., n }. Hãy duyệt tất cả các cách phân chia tập hợp đã cho thành các tập hợp con. Ví dụ { 1, 2, 3, 4 } có thể phân chia thành ba tập con là { 1, 4 }, { 2 }, { 3 }.

Ta mô tả thuật toán phân chia theo quy nạp. Giả sử tập { 1, 2, . . ., n-1 } có tất cả các cách phân chia là { B_1, B_2, . . ., B_k }. khi đó tất cả các cách phân chia của tập { 1, 2, . . ., n } sẽ là

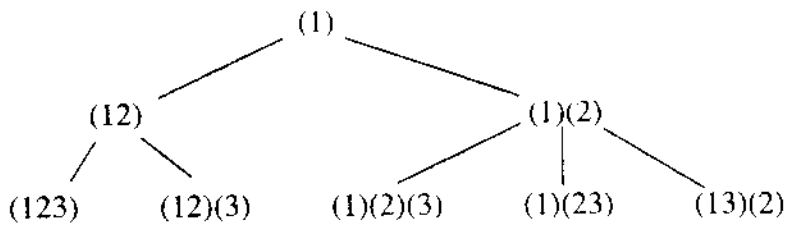
$B_1 \cup \{ n \}, B_2, \ldots, B_k$

$B_1, B_2 \cup \{ n \}, \ldots, B_k$ . . . . . .

$B_1, B_2, \ldots, B_k \cup \{ n \}$

$B_1, B_2, \ldots, B_k, \{ n \}$

Lúc đầu tập có một phần tử { 1 } chỉ có một cách phân chia là B = { 1 }.Việc xây dựng các cách phân chia tập { 1, 2, 3 } có thể mô tả bằng một cây (liệt kê các cách phân chia được cho ở hàng cuối cùng của cây):



Chương trình **PhanTap.pas**: duyệt mọi cách phân chia một tập hợp thành các tập hợp con.

```
const nmax=10;
type mang=array[1..nmax] of byte; mangb=array[1..nmax] of boolean;
var khoi,sau,truoc: mang; tien: mangb; i,j,n,k,d: integer;  f: text;

procedure InPc;
label 11,12;
begin write(f,'Phep chia thu ',d:3,' la: ');
for i:=1 to n do
  begin for j:= 1 to n do
          if khoi[j]=i then begin write(f,'('); goto 11; end;
        goto 12;
        11: for j:=1 to n do if khoi[j]=i then write(f,j:3);
        write(f,')   ');
        12:
  end; writeln(f);
end;

BEGIN assign(f,'chiatap.kqu'); rewrite(f);
write('Vao n = '); readln(n);
for i:=1 to n do begin khoi[i]:=1; tien[i]:=True end; sau[1]:=0;
d:=1; InPc; j:=n; (* j la phan tu tich cuc *)
while j > 1 do
BEGIN k:=khoi[j];
      if tien[j] then (* j chuyen dong theo chieu tien *)
         begin if sau[k]=0 then begin sau[k]:=j; truoc[j]:=k;
                                      sau[j]:=0 end;
               if sau[k]>j then begin truoc[j]:=k; sau[j]:=sau[k];
                                      truoc[sau[j]]:=j; sau[k]:=j
                                end;
               khoi[j]:=sau[k]
         end
      else                          (* j chuyen dong theo chieu lui *)
         begin khoi[j]:= truoc[k];
               if k=j then (* j lap nen khoi co 1 phan tu *)
                  if sau[k]=0 then sau[truoc[k]]:=0
                     else begin sau[truoc[k]]:=sau[k];
                                truoc[sau[k]]:=truoc[k] end
         end;
      d:=d+1;  InPc; j:=n;
      while (j>i) and ((tien[j] and (khoi[j]=j)) or
                       (not tien[j] and (khoi[j]=1)) ) do
            begin  tien[j]:= not tien[j]; j:= j-1 end;
END;
close(f); write('An Enter de ket thuc chuong trinh'); readln  END.
```

Kết quả chạy chương trình với n = 4
Phép chia thứ 1 là : (1234)
Phép chia thứ 2 là : (123) (4)
Phép chia thứ 3 là : (12) (3) (4)
Phép chia thứ 4 là : (12) (34)
Phép chia thứ 5 là : (124) (3)
Phép chia thứ 6 là : (14) (2) (3)
Phép chia thứ 7 là : (1) (24) (3)
Phép chia thứ 8 là : (1) (2) (34)
Phép chia thứ 9 là : (1) (2) (3) (4)
Phép chia thứ 10 là : (1) (23) (4)
Phép chia thứ 11 là : (1) (234)
Phép chia thứ 12 là : (14) (23)
Phép chia thứ 13 là : (134) (2)
Phép chia thứ 14 là : (13) (24)
Phép chia thứ 15 là : (13) (2) (4)

5. Duyệt mọi cách phân chia một số nguyên dương thành tổng những số nhỏ hơn

Bài toán: có bao nhiêu cách phân chia số tự nhiên n thành tổng các số tự nhiên nhỏ hơn: n = a[1] + a[2] + . . . + a[k], trong đó k, a[1], a[2], ... a[k] > 0. Ta sẽ sắp xếp các cách phân chia theo thứ tự từ vựng giảm dần. Cách phân chia đầu tiên chính là n, cách phân chia cuối cùng là tổng của n số 1. Ví dụ số 7 có tất cả các cách phân chia thành tổng các số tự nhiên không lớn hơn 7 là: 7, 6 1, 5 2, 5 1 1, 4 3, 4 2 1, 4 1 1 1, 3 3 1, 3 2 2, 3 2 1 1, 3 1 1 1 1, 2 2 2 1, 2 2 1 1 1, 2 1 1 1 1 1, 1 1 1 1 1 1 1.

Như vậy bài toán quy về vấn đề sau. Giả sử cho một cách phân chia

$$n = a[1] + a[2] + \ldots + a[t-1] + a[t] + a[t+1] + \ldots + a[k],$$

trong đó a[t] > 1, a[t+1] = a[k] = 1. Hãy tìm cách phân chia trực tiếp theo sau cách phân chia này.

Ký hiệu h = a[t] -1, s = a[t] + a[t+1] + . . . + a[k]. Khi đó cách phân chia tiếp theo sẽ là:

$$n = a[1] + a[2] + \ldots + a[t-1] + \underbrace{h + \ldots + h}_{(s \text{ div } h) \text{ lần}} + (s \bmod h),$$

Trong lập trình ta dùng hai mảng S[d] và R[d] đều có d phần tử. Các số hạng khác nhau của cách phân chia chứa trong mảng S. Số lần xuất hiện của số hạng S[i] trong cách phân chia là R[i]. Ví dụ, ứng với cách phân chia 7 = 3 + 2 + 2 ta có S ={3,2}. R = {1,2}. Chương trình **ChiaSo.pas**:

```
const nmax=20;
type mang=array[1..nmax] of integer;
var s,r: mang; i,h,d,n,dem,sum: integer; f: text;

begin assign(f,'chiaso.kqu'); rewrite(f);
write(' n = '); readln(n); s[1]:=n;r[1]:=1; d:=1; dem:=1;
writeln(f,'Cach chia ',dem:4,' la : ');
for i:=1 to d do write(f,s[i]:5); writeln(f);
for i:=1 to d do write(f,r[i]:5); writeln(f);
```

```
while s[1]>1 do
begin sum:=0;
   if s[d]=1 then begin sum:=sum+r[d]; d:=d-1; end;
   sum:=sum+s[d]; r[d]:=r[d]-1; h:=s[d]-1; if r[d]>0 then d:=d+1;
   s[d]:=h; r[d]:=sum div h; h:= sum mod h;
   if h<>0 then begin d:=d+1; s[d]:=h; r[d]:=1; end;
   dem:=dem+1; writeln(f,'Cach chia ',dem:4,' la : ');
   for i:=1 to d do write(f,s[i]:5); writeln(f);
   for i:=1 to d do write(f,r[i]:5); writeln(f);
end;  write('An Enter de ket thuc '); readln;
end.
```

Các cách phân chia của số 7 được cho trong tệp **ChiaSo.kqu** như sau:

Cách chia 1 là :	Cách chia 6 là:	Cách chia 11 là:
7	4 2 1	3 1
1	1 1 1	1 4
Cách chia 2 là :	Cách chia 7 là :	Cách chia 12 là :
6 1	4 1	2 1
1 1	1 3	3 1
Cách chia 3 là :	Cách chia 8 là :	Cách chia 13 là :
5 2	3 1	2 1
1 1	2 1	2 3
Cách chia 4 là :	Cách chia 9 là :	Cách chia 14 là :
5 1	3 2	2 1
1 2	1 2	1 5
Cách chia 5 là :	Cách chia 10 là :	Cách chia 15 là :
4 3	3 2 1	1
1 1	1 1 2	7

6. Thuật toán quay lui dùng đệ quy

Bài toán đặt ra là cần xây dựng các bộ giá trị (x[1], x[2], . . ., x[n]). Giả sử ta đã xác định được k-1 thành phần đầu tiên: x[1], x[2], . . ., x[k-1], bây giờ ta cần xác định x[k] bằng cách duyệt tất cả các khả năng mà x[k] có thể nhận giá trị (đánh số các khả năng từ 1 đến n_k). Với mỗi khả năng j ta kiểm tra xem nó có chấp nhận được hay không, hai trường hợp xảy ra:

- Nếu chấp nhận khả năng j thì xác định x[k] theo j, sau đó nếu k = n thì ta được một bộ giá trị (x[1], x[2], . . ., x[n]), nếu k < n thì ta lại tiến hành xác định x[k+1].

- Nếu thử tất cả các khả năng mà không có khả năng nào được chấp nhận thì quay lại bước trước để xác định x[k-1].

Thủ tục dạng đệ quy tiện dùng để lập trình cho thuật toán này, dưới đây là mẫu các thủ tục:

```
Procedure Test(k: integer);
var j : integer ;
Begin  for  j := 1 to n do
       if < chấp nhận j > then
          begin  < xác định x[k] theo j >
                 if k = n then < ghi nhận bộ giá trị mới >
                          else Test(k+1);
```

```
            end ;
    end;
```

Điểm quan trọng của thuật toán ở đây là cần đưa ra danh sách các khả năng mà x[k] có thể nhận giá trị, xác định giá trị của biểu thức logic < chấp nhận j >. Giá trị này ngoài việc phụ thuộc j còn phụ thuộc vào việc đã chọn các khả năng tại k-1 bước trước. Vì thế cần ghi nhớ trạng thái mới của quá trình tìm kiếm sau khi < xác định x[k] theo j > và trả lại trạng thái cũ sau lời gọi Test(k+1). Dưới đây là các chương trình sử dụng thuật toán quay lui.

Chương trình **HoanVi2.pas**: duyệt tất cả các hoán vị của tập { 1, 2, 3, . . ., n }. Một hoán vị được biểu diễn dưới dạng (p[1], p[2], . . ., p[n]), trong đó p[i] nhận giá trị từ 1 đến n và p[i] ≠ p[j] với i ≠ j . Thành phần p[k] có khả năng nhận các giá trị từ 1 đến n, trong đó giá trị j được dùng nếu nó chưa được dùng. Vì thế cần phải ghi nhớ đối với mỗi giá trị j xem nó đã được dùng hay chưa. Điều này được thực hiện nhờ mảng logic b, lúc đầu các b[j] đều được gán là True (tất cả các j đều được dùng). Mỗi lần gán j cho p[k], ta cần gán b[j] là False. Mỗi lần ghi lại trạng thái mới hay gọi hàm đệ quy HoanVi(k+1) ta cũng phải gán lại b[j] là True để trả lại trạng thái cũ. Các trạng thái này được ghi nhận nhờ một biến trạng thái.

```
var i, n: integer;
    p: array[1..20] of integer;
    b: array[1..20] of boolean; d: word;

procedure HoanVi(k: integer);
var j,i1:integer;
begin
for j:=1 to n do
  if b[j] then
     begin p[k]:=j; b[j]:= False;
           if k=n then begin  d:=d+1; write('Hoan vi ',d:4,' : ');
                              for i1:=1 to n do write(p[i1]:3);
                              writeln; if d mod 23=0 then readln;
                       end
                  else HoanVi(k+1);
           b[j]:= True;
      end;
end;
Begin write('n = '); readln(n);
for i:=1 to n do b[i]:=True; d:=0;
HoanVi(1);
write('An Enter de ket thuc'); readln;
End.
```

Chương trình **TapCon2.pas**: duyệt các tập con của một tập hợp (hay duyệt các dãy nhị phân độ dài n). Mỗi dãy nhị phân được biểu diễn dưới dạng (b[1], b[2], . . . , b[n]), trong đó b[k] có hai khả năng nhận giá trị 0 hay 1, cả hai khả năng này đều chấp nhận mà không phải thoả mãn điều kiện gì, vì thế bài toán không cần biến trạng thái.

```
var n:integer;
    b: array[1..20] of 0..1; d: word;
procedure TapCon(k: integer);
var j,i1: integer;
begin
```

```
for j:=0 to 1 do
  begin b[k]:=j;
        if k=n then begin  d:=d+1; write('Xau ',d:4,' : ');
                           for i1:=1 to n do write(b[i1]:2);
                           writeln; if d mod 23=0 then readln;
                    end  else TapCon(k+1);
  end;
end;
begin write('n = '); readln(n);
d:=0; TapCon(1);
write('An Enter de ket thuc'); readln
end.
```

Chương trình **ToHop2.pas**: liệy kê các tổ hợp chập m của n phần tử { 1, 2, . . ., n}. Một tổ hợp được biểu diễn dưới dạng 1 <= c[1] < c[2] < . . . < c[m] <= n. Từ đó suy ra các khả năng c[k] có thể nhận giá trị là từ c[k-1] + 1 đến n - m + k (các giá trị này mặc nhiên được chấp nhận). Để điều này đúng chơ cả trường hợp k = 1, cần thêm vào c[0] = 0.

```
var  n,m: integer;
    c: array[0..20] of integer; d: word;

procedure ToHop(k: integer);
var j,i1:integer;
begin for j:=c[k-1]+1 to n-m+k do
    begin c[k]:=j;
      if k=m then begin  d:=d+1; write('To hop ',d:4,' : ');
                  for i1:=1 to m do write(c[i1]:3); writeln;
                  if d mod 23=0 then readln;
                 end else ToHop(k+1);
    end;
end;

Begin write('n, m = '); readln(n,m); c[0]:=0; d:=0;
ToHop(1);
write('An Enter de ket thuc'); readln;
End.
```

Chương trình **ChinhHop.pas**: liệt kê tất cả các chỉnh hợp chập m của n phần tử {1, 2, ..., n}. Một chỉnh hợp chập m của n phần tử là một bộ sắp thứ tự gồm m phần tử lấy ra từ n phần tử đã cho. Tổng số chỉnh hợp chập m của n là n! / (n - m)!

```
var i, n,m: integer;
    p: array[1..20] of integer;
    b: array[1..20] of boolean; d: word;

procedure ChinhHop(k: integer);
var j,i1:integer;
begin
for j:=1 to n do
  if b[j] then
     begin p[k]:=j; b[j]:= False;
           if k=m then begin  d:=d+1; write('Chinh hop ',d:4,' : ');
                              for i1:=1 to m do write(p[i1]:3); writeln;
                              if d mod 23=0 then readln;
                       end else ChinhHop(k+1);
           b[j]:= True;
```

```
    end;
end;
Begin write('n, m = '); readln(n,m);
for i:=1 to n do b[i]:=True; d:=0;
ChinhHop(1);
write('An Enter de ket thuc'); readln;
end.
```

Bài tập

1. Viết chương trình tìm tất cả các hoán vị của chữ COMPUTER.

2. Cho véc tơ B = (b_1, b_2, . . ., b_n) cố định. Ký hiệu P = (p_1, p_2, . . ., p_n) là một hoán vị của (a_1, a_2, . . ., a_n). Tìm hoán vị P sao cho cực tiểu tổng $p_1b_1 + p_2b_2 + \ldots + p_nb_n$.

3. Có 8 bạn: Dũng, Cường, Hoàng, Trung, Lan, Phương, Huệ, Nhung cùng ngồi dự liên hoan theo một bàn tròn. Hãy tìm mọi cách sắp xếp sao cho thoả mãn các điều kiện: Huệ ngồi cạnh Nhung, Hoàng ngồi cạnh Trung, Cường ngồi cạnh Hoàng, Huệ không ngồi cạnh Phương, Lan ngồi bên phải Dũng.

4. Cho dãy số a_1, a_2, . . ., a_n. Hãy tìm k phần tử trong dãy số này để biểu thức sau đạt giá trị lớn nhất: $SIN(a_{i1} + a_{i2} + \ldots + a_{ik})$

5. Cho n vật với trọng lượng là p_1, p_2, . . . , p_n kilogam và có giá trị là c_1, c_2, . . ., c_n. Hãy chọn ra một số đồ vật sao cho tổng khối lượng của chúng không vượt quá 30 kilogam và tổng giá trị của chúng là lớn nhất.

6. Cho n quả cân có khối lượng lần lượt là p_1, p_2, . . ., p_n và một cái cân có hai đĩa. Tìm tất cả các cách đặt lên hai đĩa cân một số quả cân nào đó từ n quả cân trên sao cho cân thăng bằng.

7. Nhập từ bàn phím số nguyên dương N <= 30000. Hãy tìm cách biểu diễn N thành tổng các số nguyên dương A_1, A_2, . . ., A_k sao cho tích các số này là lớn nhất có thể được.

8. Có N gói kẹo, N <= 200, các gói kẹo được đánh số từ 1 đến N, gói kẹo thứ i có A_i cái kẹo, các số A_1, A_2, . . ., A_n đều nguyên dương và không quá 200. Hãy thông báo ra màn hình một cách chia các gói kẹo làm hai nhóm sao cho tổng số kẹo trong các nhóm sai khác nhau ít nhất.

9. Liệt kê tất cả các cách xếp n quân hậu trên bàn cờ n hàng n cột sao cho chúng không ăn được lẫn nhau.

Chương 13

Đồ thị hữu hạn

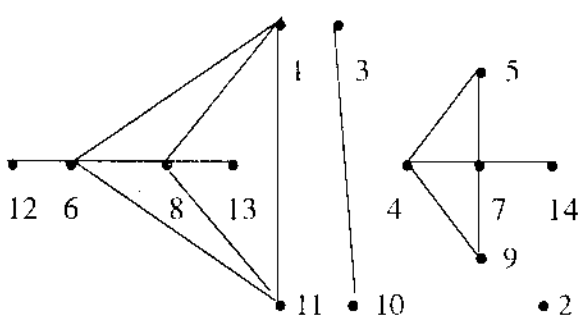
Chương này trình bày về đồ thị có hướng và đồ thị vô hướng, tính số thành phần liên thông của đồ thị, tìm kiếm theo chiều sâu và tìm kiếm theo chiều rộng trên đồ thị, tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị, tìm cây bao trùm ngắn nhất.

1. Đồ thị và tính liên thông

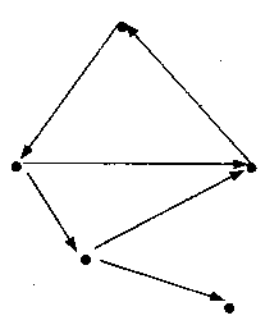
Một **đồ thị vô hướng** G gồm hai tập hợp hữu hạn V và E, đồ thị G được ký hiệu là G = (V, E). V gọi là tập đỉnh của đồ thị, không giảm tổng quát ta có thể xem n đỉnh của G là tập n số nguyên dương đầu tiên V = {1, 2, . . ., n}. E là tập hợp con các tích đề các V x V mà nếu (i, j) thuộc E thì (j, i) cũng thuộc E và ta đồng nhất (i, j) với (j, i). Mỗi phần tử của E gọi là một cạnh của đồ thị. Nếu (i, j) là một cạnh ta nói cạnh đó nối đỉnh i với đỉnh j, ta cũng nói hai đỉnh i và j là kề nhau.

Một **đồ thị có hướng** G gồm hai tập hợp hữu hạn V và E và ký hiệu là G = (V, E). V là tập đỉnh của đồ thị. E là tập hợp các cung của đồ thị, nó là tập con của tích đề các V x V. Nếu (i, j) là một cung thì ta nói (i, j) là cung đi từ đỉnh i tới đỉnh j. Trong đồ thị có hướng cung (i, j) khác với cung (j, i).

Trong thực tế ta thường gặp đồ thị vừa có cạnh (vô hướng) vừa có cung (có hướng), chẳng hạn như đường giao thông trong thành phố. Để chuyển một đồ thị vô hướng thành có hướng ta có thể thay mỗi cạnh vô hướng bằng hai cung có hướng ngược nhau cùng nối liền hai đỉnh. Biểu diễn hình học một đồ thị: dùng dấu chấm để biểu thị đỉnh, số ở cạnh đỉnh là tên đỉnh, cạnh nối đỉnh i và đỉnh j biểu thị bằng một đường nối hai đỉnh, cung nối hai đỉnh biểu thị bằng mũi tên. Hình 1 là đồ thị vô hướng, Hình 2 là đồ thị có hướng.



Hình 1



Hình 2

Biểu diễn đồ thị trên máy tính bằng ma trận kề. Ta dùng một mảng gồm n hàng và n cột a[1..n,1..n], trong đó n là số đỉnh của đồ thị. Nếu từ đỉnh i tới đỉnh j không có cạnh hay cung nào thì a[i,j] = 0, còn nếu có cạnh hay cung đi từ i đến j thì a[i,j]=1. Ma trận kề của một đồ thị vô hướng là một ma trận đối xứng, ma trận kề của một đồ thị có hướng nói chung là không đối xứng.

Một **đường đi** từ đỉnh u tới đỉnh v là một dãy cạnh $e_1, e_2, \ldots, e_k$ mà $e_i = (u_i, u_{i+1})$, $1 <= i <= k-1$, $u_1 = u$ và $u_k = v$. Một đường đi khép kín (tức là u = v) gọi là một **chu**

trình. Đồ thị gọi là liên thông nếu với mọi cặp đỉnh u và v của nó ta luôn có đường đi từ u tới v.

Đồ thị G gọi là không liên thông nếu tồn tại một cặp đỉnh u và v mà không có đường đi nào nối chúng. Khi đó đồ thị G chia thành các thành phần liên thông. Bài toán đặt ra là cho một đồ thị vô hướng G, hãy cho biết G có bao nhiêu thành phần liên thông và mỗi thành phần liên thông gồm các đỉnh nào.

Chương trình **LienTh.pas** giải quyết bài toán trên. Dữ liệu nhập vào theo một tệp văn bản LienTh.dat: dòng đầu ghi n (số đỉnh của đồ thị) và m (tổng số cạnh), tiếp theo là m cặp (i, j) ứng với mỗi cạnh nối đỉnh i và đỉnh j. Tệp dữ liệu ứng với đồ thị ở Hình 1 là:

```
14 15
1 6    1 8    1 11   3 10   4 5    4 7    4  9
5 7    6 8    6 11   6 12   7 9    7 14   8 11   8 13
```

Hàm Lienth() cho số thành phần liên thông của đồ thị. Kết thúc chương trình mảng p[1..n] cho biết mỗi đỉnh i thuộc thành phần liên thông thứ p[i]. Kết quả chạy chương trình: thành phần liên thông thứ 1 gồm các đỉnh 1, 6, 8, 11, 12, 13; thành phần liên thông thứ 2 gồm đỉnh 2; thành phần liên thông thứ 3 gồm 3, 10; thành phần liên thông thứ 4 gồm 4, 5, 7, 9, 14.

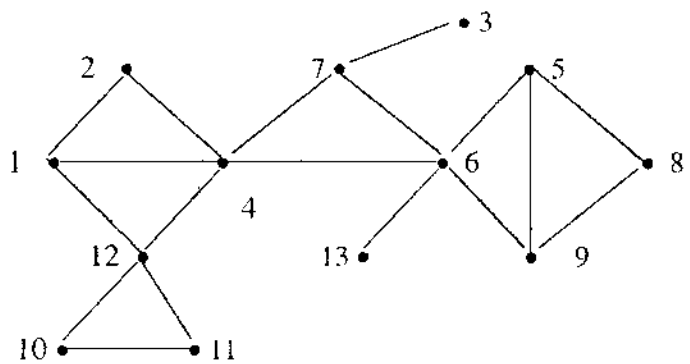
```
uses crt;
const nmax=30;
type mang1= array[1..nmax,1..nmax] of byte;
     mang2= array[1..nmax] of byte;
var c: mang1; p: mang2; i,j,n,m,d: integer; f: text;

function lienth(n: integer; c: mang1; var p: mang2): integer;
var i,j,d,k,l: integer;
begin  for i:=1 to n do p[i]:=0;  d:=0;
for i:=1 to n do
 begin  if p[i]=0 then
     begin  inc(d); p[i]:=d;
       for k:=1 to n do for j:=1 to n do for l:=1 to n do
          if (p[j]=0) and (p[l]=d) and (c[l,j]=1) then p[j]:=d;
     end;
 end; lienth:=d;
end;

Begin clrscr;  assign(f,'LienTh.dat'); reset(f); readln(f,n,m);
for i:=1 to n do for j:=1 to n do c[i,j]:=0;
for d:=1 to m do  begin read(f,i,j); c[i,j]:= 1 end;  close(f);
i:= lienth(n,c,p); writeln('So thanh phan lien thong = ',i);
writeln('Vec to P : ');
for j:=1 to n do  writeln('    p[',j:2,'] = ',p[j]); readln;
End.
```

2. Tìm kiếm theo chiều sâu

Bài toán duyệt qua tất cả các đỉnh của đồ thị vô hướng G (tại mỗi đỉnh này ta có thể làm một công việc gì đó) là một bài toán quan trọng, nó là cơ sở cho nhiều thuật toán khác. Chẳng hạn nó có thể dùng để tìm số thành phần liên thông của đồ thị, xét

xem có tồn tại đường đi giữa hai đỉnh bất kỳ của đồ thị hay không. Có hai phương pháp duyệt các đỉnh của đồ thị: tìm kiếm theo chiều sâu và tìm kiếm theo chiều rộng.

Tư tưởng của phương pháp tìm kiếm theo chiều sâu như sau. Ta bắt đầu tìm kiếm với một đỉnh cố định v_0 nào đó. Sau đó lựa chọn một đỉnh tuỳ ý v là kề với v_0, và lặp lại quá trình này từ v. Tổng quát, giả sử ta đang ở đỉnh v, nếu tồn tại một đỉnh mới u chưa xét kề với v thì ta xét đỉnh này, ta lại tiếp tục tìm kiếm từ đỉnh u này. Nếu không tồn tại đỉnh mới chưa xét nào kề với v thì chúng ta nói đỉnh v đã được dùng, sau đó chúng ta quay lại đỉnh mà từ đó chúng ta đã tới v, và ta tiếp tục quá trình (nếu $v = v_0$ thì việc tìm kiếm kết thúc). Nói cách khác việc tìm kiếm theo chiều sâu từ đỉnh v dựa trên việc tìm kiếm theo chiều sâu từ tất cả các đỉnh mới kề với đỉnh v.



Hình 3

Xét đồ thị trong Hình 3, gọi n là số đỉnh của đồ thị. Các đỉnh được đánh số từ 1 đến n theo thứ tự tuỳ ý. Trong chương trình dùng các mảng:

- Mảng moi[1..n] kiểu nguyên, moi[i] = 1 nếu đỉnh i chưa được xét (còn là mới) và bằng 0 nếu đã xét. Lúc đầu tất cả các phần tử của nó đều bằng 1.
- Mảng s[1..n] có nghĩa: đỉnh i có s[i] đỉnh kề.
- Mảng p[1..n,1..smax] có nghĩa: đỉnh thứ i có s[i] đỉnh kề là p[i,1], p[i,2], .. ., p[i,s[i]]. Như vậy đồ thị được biểu diễn dưới dạng danh sách kề.
- Mảng stack[1..n] dùng để chứa các đỉnh trong quá trình xét, nó có cấu trúc kiểu ngăn xếp.

Thuật toán chi tiết tìm kiếm theo chiều sâu bắt đầu từ đỉnh v như sau:

```
stack = rỗng; Đưa v vào stack; moi[v] = 0;
Thông báo đỉnh v đã xét (đã đi tới);
while (stack != rỗng) {
    u = phần tử ở đỉnh stack;
    Xét tất cả các đỉnh j kề với u xem có đỉnh nào chưa xét không ?
   - Nếu có đỉnh j chưa xét thì: k = j; đưa k vào stack; thông báo đã xét đỉnh k; moi[k] = 0;
   - Nếu mọi đỉnh j kề với u đều đã xét thì huỷ phần tử trên đỉnh stack;   }
```

Nếu đồ thị chỉ có một thành phần liên thông và ta xuất phát từ một đỉnh v bất kỳ thì thuật toán trên sẽ duyệt qua tất cả các đỉnh. Nếu đồ thị có nhiều thành phần liên thông và ta xuất phát từ một đỉnh v thì ta chỉ duyệt được tất cả các đỉnh của một thành phần liên thông chứa đỉnh v.

Chương trình **TkcSau.pas** thực hiện thuật toán đã nêu. Số liệu nhập vào từ một tệp văn bản TkcSau.dat. Với đồ thị như ở Hình 3 tệp số liệu có dạng:

```
13
3 2 1 5 3 5 3 2 3 2 2 4 1
2 4 12
```

```
1 4
7
1 2 6 7 12
6 8 9
4 5 7 9 13
3 4 6
5 9
5 6 8
11 12
10 12
1 4 10 11
6
```

Dòng đầu tiên chứa n (số đỉnh). Dòng thứ hai chứa mảng s[1..n]. Còn n dòng tiếp theo: mỗi dòng chứa các đỉnh kề của đỉnh thứ i.

Khi xuất phát từ đỉnh 1 ta nhận được kết quả tìm kiếm các đỉnh theo chiều sâu lần lượt như sau: 1. 2. 4. 6. 5. 8, 9, 7, 3, 13. 12, 10, 11.

```
uses crt;
const nmax=20; smax=10;
type mang1 = array[1..nmax,1..smax] of integer;
     mang2 = array[1..nmax] of boolean;
     mang3 = array[1..nmax] of integer;
var moi: mang2; p : mang1;  s: mang3;
    i,j,n: integer; f: text;

procedure TimSau(v: integer);
var stack: mang3; k,i,top,it,u: integer;
begin  for i:=1 to n do stack[i]:=0; stack[1]:= v; Top := 1;
write('Xuat phat tu dinh ',v); readln; moi[v]:= false; it:=1;
while Top<> 0 do
  begin   writeln('==================================');
     writeln('Buoc lap ',it,' : top = ',top); write('Mang Stack: ');
     for i:=1 to top do write(stack[i],'  '); writeln;
     write('Mang Moi : ');
     for i:=1 to n do write(moi[i],'  '); writeln;
     k:=0; u:= stack[top];
     for j:= 1 to s[u] do if moi[p[u,j]] then
               begin k:= p[u,j]; break; end;
     writeln('Chon K = ',k);
     if k<>0 then
       begin inc(top); stack[top]:= k;
         writeln('Dua vao stack : ',stack[top]);
         write('Toi dinh moi ',k); readln;  moi[k]:= false;
       end
     else begin  writeln('Lay khoi stack dinh: ',stack[top]);
                 readln; stack[top]:=0; dec(top)
          end;
     inc(it);
  end;
end;

Begin clrscr; assign(f,'TkcSau.dat'); reset(f);
readln(f,n); for i:=1 to n do  read(f,s[i]);
for i:=1 to n do for j:=1 to s[i] do  read(f,p[i,j]); close(f);
write('Ma ng S:'); for i:=1 to n do write(s[i],'  '); writeln;
```

```
for i:=1 to n do
 begin write('Dinh thu ',i,'..... ');
      for j:=1 to s[i] do write(p[i,j],'   ');  writeln;
 end;
for i:=1 to n do Moi[i]:= true; TimSau(1);
write('An Enter de ket thuc! '); readln;
end.
```

3. Tìm kiếm theo chiều rộng

Ta vẫn xét đồ thị cho trong Hình 3, các mảng **moi, p, s, stack** vẫn có ý nghĩa như ở mục trước. Thuật toán tìm kiếm theo chiều rộng xuất phát từ đỉnh v có thể mô tả như sau:

stack = rỗng; Đưa v vào stack; moi[v] = 0;
Thông báo đỉnh v đã xét (đã đi tới);
while (stack != rỗng) {
- Lấy phần tử ở đỉnh stack ra khỏi stack và đưa vào biến k;
- Thông báo đã xét đỉnh k;
- Với mỗi đỉnh j kề với đỉnh k, nếu moi[j]=1 thì ta đưa j vào stack, đồng thời gán moi[j]=0; }

Phương pháp gọi là tìm kiếm theo chiều rộng vì mỗi lần ta đưa vào stack tất cả các đỉnh chưa xét kề với đỉnh k đã xét. Nếu ta xuất phát từ một đỉnh v của đồ thị thì ta duyệt được tất cả các đỉnh của một thành phần liên thông chứa đỉnh v này.

Chương trình **TkcRong.pas** thực hiện thuật toán trên.

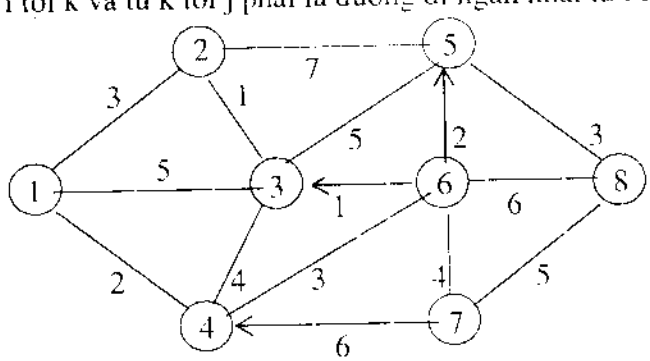
```
uses crt;
const nmax=50; smax=10;
type mang1 = array[1..nmax,1..smax] of integer;
     mang2 = array[1..nmax] of boolean;
     mang3 = array[1..nmax] of integer;
var moi: mang2; p : mang1;  s: mang3; i,j,n,tg: integer; f: text;

procedure TimRong(v: integer);
var stack: mang3; i,top,it,u,k: integer;
begin for i:=1 to n do stack[i]:=0; stack[1]:= v; Top := 1;
moi[v]:= false; it:=1;
while Top<> 0 do
  begin  writeln('========================');
     writeln('Buoc lap ',it,' : top = ',top); write('Mang Stack: ');
     for i:=1 to top do write(stack[i],'  '); writeln;
     write('Mang Moi : ');
     for i:=1 to n do write(moi[i],'  '); writeln;
     writeln('Lay ra khoi stack dinh: ',stack[top]);
     k:= stack[top]; stack[top]:=0; dec(top);
     writeln('Toi dinh moi ',k); readln;
     for j:= 1 to s[k] do
       begin u:= p[k,j];
         if moi[u] then
          begin inc(top); stack[top]:= u;
           writeln('Dua vao stack : ',stack[top]); moi[u]:= false;
          end;
       end;
     inc(it);
  end;
```

```
end;
Begin clrscr; assign(f,'TkcRong.dat'); reset(f); readln(f,n);
for i:=1 to n do  read(f,s[i]);
for i:=1 to n do for j:=1 to s[i] do  read(f,p[i,j]); close(f);
write('Ma ng S:'); for i:=1 to n do write(s[i],'  '); writeln;
for i:=1 to n do  begin write('Dinh thu ',i,'..... ');
                        for j:=1 to s[i] do write(p[i,j],'   ');
                        writeln;
                  end;
for i:=1 to n do Moi[i]:= true; TimRong(1); write('Xong... ');
readln;
end.
```

Số liệu được vào từ tệp văn bản. Với ví dụ ở Hình 3, tệp dữ liệu TkcRong.dat có dạng như tệp TkcSau.dat ở mục trên. Khi tìm kiếm xuất phát từ đỉnh 1 ta nhận được kết quả tìm kiếm các đỉnh lần lượt như sau: 1, 12, 11, 10, 4, 7, 3, 6, 13, 9, 8, 5, 2.

4. Tìm đường đi ngắn nhất giữa hai đỉnh

Cho đồ thị có hướng G = (V, E) với tập các đỉnh V = {1, 2, . . ., n} và ma trận độ dài các cung là c[1..n,1..n], trong đó c[i,j] là độ dài cung đi từ đỉnh i tới đỉnh j, nếu không có cung đi từ i tới j thì ta cho c[i,j] bằng một số rất lớn (ví dụ lớn hơn tổng độ dài tất cả các cung), c[i,i] = 0 với mọi i = 1, 2, . . ., n (Hình 4). Thuật toán Floyd thiết kế theo kỹ thuật quy hoạch động cho phép tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị. Nếu một đỉnh k nằm trên đường đi ngắn nhất từ đỉnh i tới đỉnh j thì đoạn đường từ i tới k và từ k tới j phải là đường đi ngắn nhất từ i tới k và từ k tới j tương ứng.



Hình 4

Ta sử dụng ma trận a[1..n,1..n] để lưu độ dài đường đi ngắn nhất giữa mọi cặp đỉnh, ban đầu đặt a[i,j] = c[i,j] với mọi (i, j), tức là a chứa độ dài đường đi trực tiếp không qua đỉnh nào cả. Sau đó ta tiến hành n lần lặp, sau lần lặp thứ k ma trận a sẽ chứa độ dài các đường đi ngắn nhất chỉ đi qua các đỉnh thuộc {1, 2, . . ., k}, sau n lần lặp ta nhận được a chứa độ dài các đường đi ngắn nhất.

Ta ký hiệu a_k là ma trận a sau lần lặp thứ k, tức là $a_k[i,j]$ là độ dài đường đi ngắn nhất từ i đến j chỉ đi qua các đỉnh thuộc {1, 2, . . ., k}, công thức tính $a_k[i,j]$:

$$a_k[i,j] = \min \{ a_{k-1}[i,j], a_{k-1}[i,k] + a_{k-1}[k,j] \}$$

Các phần tử trên đường chéo của các ma trận a_k luôn bằng 0. Để lưu vết của đường đi ngắn nhất ta dùng một ma trận nguyên p[1..n,1..n], lúc đầu p[i,j] = 0 với mọi (i,j), sau đó p[i,j] lưu đỉnh k nếu đường đi ngắn nhất từ i đến j được tìm ra bởi thuật toán Floyd đi qua đỉnh k.

Chương trình **Ddnn.pas** lập theo thuật toán Floyd. Dữ liệu vào theo tệp văn bản Ddnn.dat, với đồ thị ở Hình 4 tệp dữ liệu có dạng:

```
8
0    3    5    2    1000 1000 1000 1000
3    0    1    10   7    1000 1000 1000
5    1    0    4    5    1000 1000 1000
2    1000 4    0    1000 3    1000 1000
1000 7    5    1000 0    1000 1000 3
1000 1000 1    3    2    0    4    6
1000 1000 1000 6    1000 4    0    5
1000 1000 1000 1000 3    6    5    0
```

Dòng đầu tiên là n (số đỉnh), n dòng tiếp theo lưu các dòng của ma trận độ dài c[1..n,1..n]. Nếu từ đỉnh i tới đỉnh j không có cung ta cho c[i,j] = 1000.

```
uses crt;
const  nmax=10;
var  i,j,n: integer; t: real; f: text;
     p: array[1..nmax,1..nmax] of integer;
     a: array[1..nmax,1..nmax] of real;

procedure Floyd;
var i, j, k : integer;
begin for i:=1 to n do for j:=1 to n do p[i,j] := 0;
for k:=1 to n do for i:=1 to n do for j:=1 to n do
   if (a[i,k]+a[k,j] < a[i,j]) then
       begin  a[i,j] := a[i,k] + a[k,j]; p[i,j] := k end;
end;

Procedure DDi2dinh;
var u,v,i,k, nd: integer;
    dd,g: array[1..nmax] of integer;
begin
write('Tim duong di ngan nhat tu dinh u = '); readln(u);
write('..... den dinh v = '); readln(v);
writeln('Do dai duong di ngan nhat la = ',a[u,v]:1:5);
dd[1]:= u; dd[2] := v;
nd := 2; g[1]:= p[u,v];
if (g[1]<>0) then
  begin while true do begin
      k:=0; for i:=1 to nd-1 do
                if (g[i]>0) then begin k:=i; break; end;
      if (k=0) then break;
      for i:=nd+1 downto k+2 do dd[i]:=dd[i-1]; dd[k+1]:= g[k];
      for i:=nd downto k+2 do g[i]:=g[i-1];
      g[k]:=p[dd[k],dd[k+1]];
      g[k+1]:= p[dd[k+1],dd[k+2]]; inc(nd);
                    end
  end;
writeln('Duong di ngan nhat la: ');
for i:=1 to nd do write(dd[i],' - '); writeln;
end;

Begin clrscr;
assign(f,'DDnn.dat'); reset(f);
readln(f,n);
```

```
for i:=1 to n do for j:=1 to n do
                 begin read(f,t); a[i,j]:=t end;
close(f);
writeln('So dinh cua do thi : ',n);
for i:=1 to n do begin write('Dong ',i,' : ');
                       for j:=1 to n do write(a[i,j]:1:2,' - ');
                       writeln
                  end;
readln;  Floyd;
writeln('Ma tran A[1..n,1..n] : ');
for i:=1 to n do begin write('Dong ',i,' : ');
                         for j:=1 to n do write(a[i,j]:1:2,' - ');
                         writeln;
                  end;
readln; writeln('Ma tran P[1..n][1..n] : ');
for i:=1 to n do begin write('Dong ',i,' : ');
                         for j:=1 to n do write(p[i,j],' - ');
                         writeln;
                  end; readln;
while true do begin
   writeln('--------------------------------');
   write('Co tim duong di ngan nhat khong 1/0: '); readln(j);
   if j=1 then DDi2dinh else break;
              end
End.
```

Mảng a[1..n,1..n] trong chương trình lúc đầu lưu ma trận độ dài c[1..n,1..n], sau đó lưu các ma trận a_k[1..n,1..n] của mỗi bước lặp. Thủ tục Floyd thực hiện thuật toán Floyd, kết quả tính toán ma trận a sau n bước lặp và ma trận p ứng với đồ thị trong Hình 4:

A:

0	3	4	2	7	5	9	10
3	0	1	5	6	8	12	9
4	1	0	4	5	7	11	8
2	5	4	0	5	3	7	8
9	6	5	9	0	9	8	3
5	2	1	3	2	0	4	5
8	6	5	6	6	4	0	5
11	8	7	9	3	6	5	0

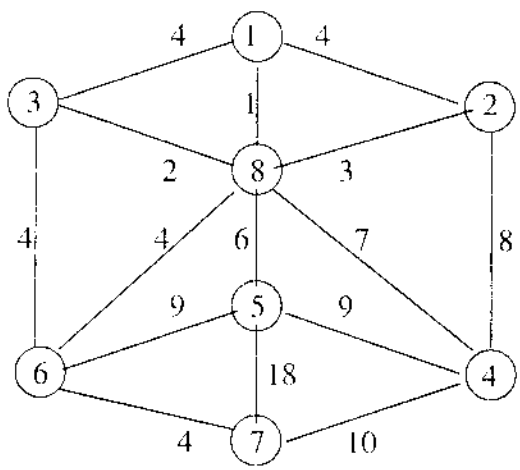
P:

0	0	2	0	6	4	6	6
0	0	0	1	3	4	6	5
2	0	0	0	0	4	6	5
0	1	0	0	6	0	6	6
3	3	0	3	0	8	8	0
3	3	0	0	0	0	0	5
4	6	6	0	6	0	0	
6	6	6	6	0	0	0	0

Thủ tục DDi2dinh nhằm xác định đường đi ngắn nhất giữa hai đỉnh qua các đỉnh nào nhờ sử dụng ma trận p. Giả sử cần tìm đường đi ngắn nhất từ đỉnh u =1 tới đỉnh v=8, tra trong bảng p ta thấy từ đỉnh 1 tới đỉnh 8 phải qua đỉnh 6, từ đỉnh 1 tới đỉnh 6 lại phải qua đỉnh 4, từ đỉnh 6 đến đỉnh 8 lại phải qua đỉnh 5. Từ đỉnh 1 tới 4, 4 tới 6, 6 tới 5, 5 tới 8 đều là đường đi trực tiếp. Vậy đường đi ngắn nhất qua các đỉnh: 1, 4, 6, 5, 8; còn độ dài là 10 cho bởi ma trận a. Đường đi từ đỉnh 8 tới đỉnh 1 có độ dài là 11 qua các đỉnh: 8, 6, 3, 2, 1.

5. Cây bao trùm ngắn nhất

Giả sử G = (V, E) là đồ thị vô hướng có n đỉnh và ma trận độ dài các cạnh là c[1..n,1..n]: độ dài cạnh nối hai đỉnh i và j là c[i,j], nếu giữa i và j không có cạnh thì c[i,j] bằng một số khá lớn, c[i,i]=0 với mọi i, c là ma trận đối xứng. T gọi là cây bao trùm của đồ thị G nếu T là đồ thị con liên thông, không có chu trình và chứa tất cả các đỉnh của G. T có n-1 cạnh. Độ dài của cây bao trùm T là tổng độ dài các cạnh tạo nên T. Bài toán đặt ra là tìm cây bao trùm T của G sao cho T có độ dài ngắn nhất. Ví dụ đồ thị ở Hình 5 có cây bao trùm ngắn nhất gồm các cạnh: (1,8) (8,3) (8,2) (8,6) (6,7) (8,5) (8,4) và tổng độ dài các cạnh là 27.



Hình 5

Ví dụ thực tế. Giả sử có n thành phố, cần xây dựng hệ thống đường xe lửa sao cho hai thành phố bất kỳ đều nối với nhau được bằng đường xe lửa và tổng độ dài của toàn bộ hệ thống đường xe lửa là ngắn nhất. Bài toán này dẫn đến bài toán tìm cây bao trùm ngắn nhất nối liền n thành phố.

Thuật toán Prim. Gọi U là tập đỉnh kề các cạnh trong T, ban đầu U chứa một đỉnh tuỳ ý của G, còn tập các cạnh T rỗng. Tại mỗi bước ta chọn cạnh (u,v) ngắn nhất sao cho u thuộc U và đỉnh v không thuộc U, rồi thêm v vào U và thêm cạnh (u,v) vào T. Điều này đảm bảo rằng sau mỗi bước T luôn luôn là một cây (không có chu trình). Ta tiếp tục phát triển cây T cho tới khi U = V, tức là T trở thành cây bao trùm.

Chương trình **CayBt.pas** thể hiện thuật toán Prim. Dữ liệu vào cho trong tệp văn bản CayBt.dat, với đồ thị ở Hình 5 tệp dữ liệu có dạng:

```
        4       4       1000    1000    1000    1000    1
4       0       1000    8       1000    1000    1000    3
4       1000    0       1000    1000    4       1000    2
1000    8       1000    0       9       1000    10      7
1000    1000    1000    9       0       9       18      6
1000    1000    4       1000    9       0       4       4
1000    1000    1000    10      18      4       0       1000
1       3       2       7       6       4       1000    0
```

Dòng đầu lưu n (số đỉnh), n dòng sau lưu ma trận độ dài c[1..n][1..n].

Mảng t lưu các cặp cạnh của cây bao trùm, hai số liên tiếp là một cạnh: (t[1], t[2]), (t[3], t[4]), (t[5], t[6]), . . . Mảng g[1..n] và d[1..n] có ý nghĩa: với mỗi đỉnh v không thuộc U thì g[v] là đỉnh u thuộc U gần đỉnh v nhất và d[v] là độ dài của cạnh nối đỉnh v không thuộc U với đỉnh g[v]. Các đỉnh u thuộc U được đánh dấu bằng cách đặt d[u] = -1. Để chọn cạnh (u,v) ngắn nhất với u thuộc U và v không thuộc U ta chỉ cần tìm min của các d[v] với v không thuộc U.

```
uses crt;
const nmax=10;
var  n,i,j,r,s,dai: integer;
     c: array[1..nmax,1..nmax] of integer;
     t: array[1..2*nmax] of integer; f: text;

procedure  Prim;
var i, j, u, v, k, min: integer;
    g,d: array[1..nmax] of integer;
begin
for i:=1 to 2*(n-1) do t[i] := 0; j := 0;
for v:=2 to n do begin g[v]:=1; d[v] := c[1][v] end;
for i:=2 to n do begin  min := 32767;
   for k:=2 to n do if (d[k]>=0) and (d[k]< min) then
                    begin min := d[k]; v:=k; u := g[v]; end;
   t[j+1] := u; t[j+2] := v; j := j+2; d[v] := -1;
   for k:=2 to n do if (c[v][k] < d[k]) then
                    begin  d[k] := c[v][k]; g[k] := v; end;
                  end;
end;

Begin  clrscr;
assign(f,'CayBt.dat'); reset(f);
readln(f,n);
for i:=1 to n do for j:=1 to n do read(f,c[i,j]);
close(f);
writeln('So dinh cua do thi : ',n);
for i:=1 to n do begin write('Dong ',i,' : ');
                       for j:=1 to n do write(c[i,j],'- '); writeln
                  end;
readln; Prim;
writeln('Cac canh cua cay bao trum : '); dai := 0;
for i:=1 to n-1 do
    begin  r := t[2*(i-1)+1]; s := t[2*(i-1)+2];
           dai := dai+c[r,s];
           write('(',r,' - ',s,') ');
    end;
writeln;
writeln('Do dai cay bao trum = ',dai); readln;
End.
```

Bài tập

1. Dùng thủ tục TimSau trong Mục 2 để tìm số thành phần liên thông của một đồ thị vô hướng.

2. Cho một đồ thị vô hướng và liên thông G = (V,E), mọi đỉnh của G đều có số cạnh liên thuộc là chẵn. Hãy tìm một đường đi khép kín trong G đi qua mọi cạnh của

đồ thị, mỗi cạnh đúng một lần (gọi là chu trình Euler). Thuật toán: xuất phát từ một đỉnh u nào đó của G và đi theo các cạnh của đồ thị một cách tuỳ ý chỉ cần tuân theo hai quy tắc sau: xoá bỏ cạnh đã đi qua đồng thời xoá cả những đỉnh cô lập tạo thành, tại mỗi bước ta chỉ đi qua cầu khi không còn cách nào khác. Cầu là một cạnh mà khi ta bỏ nó thì đồ thị trở nên không liên thông. Đỉnh cô lập là đỉnh không có cạnh nối với nó.

3. Cho một đồ thị vô hướng và liên thông G = (V,E) biểu thị bằng ma trận kề c[1..n,1..n]: nếu có cạnh từ đỉnh i tới đỉnh j thì c[i,j]=1, trái lại c[i,j]=0. Hãy tìm tất cả các đường đi khép kín trong G đi qua mọi đỉnh của đồ thị, mỗi đỉnh đúng một lần (gọi là chu trình Hamilton). Thuật toán: mỗi một đường đi qua n đỉnh ứng với một hoán vị của tập {1, 2, . . ., n}, ta chỉ cần duyệt qua mọi hoán vị của tập này và kiểm tra xem hoán vị có là một đường đi Hamilton trong G hay không.

4. Cho đồ thị G = (V,E), V gồm n đỉnh, E gồm m1 cạnh vô hướng và m2 cung có hướng. Với mỗi cạnh hay cung có một số dương c(i,j) là khoảng cách từ đỉnh i tới đỉnh j. Viết chương trình tìm đường đi ngắn nhất từ đỉnh gốc bất kỳ $i^* \in V$ tới một đỉnh đích bất kỳ $j^* \in V$ theo thuật toán gán nhãn như sau:

Bước 1. Gán nhãn cho đỉnh nguồn bằng 0: Nhan(i*) = 0. Xây dựng tập các đỉnh đã gán nhãn K = { i* }.

Bước 2. Trong số các cạnh hoặc cung đi từ đỉnh nguồn ta chọn cạnh hoặc cung có độ dài nhỏ nhất, rồi gán nhãn cho nút cuối của cạnh hay cung vừa chọn này bằng nhãn của đỉnh nguồn cộng với độ dài nhỏ nhất này. Cụ thể, chọn cạnh hay cung (i*,j1) sao cho c(i*,j1) = min{ c(i*,j): (i*,j) ∈ E}, Nhan(j1) = 0 + c(i*,j1), tập các đỉnh đã gán nhãn mới K = K ∪ {j1}.

Bước 3. Trong số các cạnh hoặc cung đi từ một đỉnh đã gán nhãn tới một đỉnh chưa gán nhãn, ta lại chọn cạnh hay cung sao cho độ dài của nó cộng với nhãn của đỉnh đã được gán nhãn là nhỏ nhất, giá trị nhỏ nhất này được gán cho đỉnh cuối của cạnh hay cung đó. Cụ thể, chọn (i2,j2) sao cho Nhan(i2) + c(i2,j2) = min { Nhan(i) + c(i,j): i ∈ K, j ∉ K, (i,j) ∈ E}, Nhan(j2) = Nhan(i2) + c(i2,j2), K = K ∪ {j2}.

Bước 4. Nếu j2 = j* thì dừng, độ dài đường đi ngắn nhất phải tìm chính là Nhan(j2). Nếu j2 ≠ j* thì ta quay lại Bước 3.

Dữ liệu được vào từ tệp văn bản. Dòng đầu gồm 3 số viết cách nhau bởi dấu cách: số cạnh vô hướng m1, tổng số cạnh vô hướng và cung có hướng m1+m2, số đỉnh n. Tiếp theo là m1 bộ ba số i, j, c(i,j) ứng với mỗi cạnh (i,j). Tiếp nữa là m2 bộ ba số i, j, c(i,j) ứng với mỗi cung (i,j).

5. Cho một đồ thị vô hướng n đỉnh biểu diễn bằng ma trận kề a[1..n,1..n] (xem Mục 1) và hai đỉnh u, v bất kỳ. Hãy tìm đường đi từ u tới v (nếu có) qua ít cạnh nhất.

Lời giải

Bài 1 chương 1 : chu vi và diện tích tam giác theo toạ độ 3 đỉnh

```
var x1,y1,x2,y2,x3,y3,c12,c13,c23,cv,p,dt: real;
begin write('x1,y1,x2,y2,x3,y3: '); readln(x1,y1,x2,y2,x3,y3);
c12 := sqrt( sqr(x1-x2)+ sqr(y1-y2) );
c13 := sqrt( sqr(x1-x3)+ sqr(y1-y3) );
c23 := sqrt( sqr(x2-x3)+ sqr(y2-y3) );
cv := c12+c13+c23; p:= cv/2;
dt := sqrt(p * (p-c12) * (p-c13) * (p-c23));
writeln('Chu vi = ',cv:1:5,' Dien tich = ',dt:1:5);
readln; end.
```

Bài 2 chương 1 : gửi tiền tiết kiệm

```
var a,s: real; t: integer;
begin write('Vao a,s,t = '); readln(a,s,t); s:= s/100;
writeln('Tong so tien = ',exp(ln(a)+t*ln(1+s)):1:5);
readln; end.
```

Bài 5 chương 1 : các hàm lượng giác của một góc theo độ phút giây

```
var d,p,g: word; rad: real;
begin
write('Vao so do cua goc (do, phut, giay): ');readln(d,p,g);
rad:=(pi/(180.0*3600.0))*(3600.0*d+60*p+g);
writeln('sin = ',sin(rad):1:5); writeln('cos = ',cos(rad):1:5);
writeln('tang = ',sin(rad)/cos(rad):1:5);
writeln('cotang = ',cos(rad)/sin(rad):1:5); readln;
end.
```

Bài 6 chương 1 : tổng các chữ số của một số

```
var so: longint; s1, s2, s3, s4, s5: integer;
begin
write('Nhap so co 5 chu so : '); readln(so);
s1 := so mod 10; s2:= (so div 10) mod 10;
s3:= (so div 100) mod 10; s4:= (so div 1000) mod 10;
s5:= so div 10000;
writeln('Tong cac chu so la : ',s1+s2+s3+s4+s5);
readln; end.
```

Bài 1 chương 2 : ba số là ba cạch của một tam giác

```
var a,b,c,p,s: real;
begin write('Vao a,b,c: '); readln(a,b,c);
if (a>0)and(b>0)and(c>0)and(a+b>c)and(a+c>b)and (b+c>a)then
begin  writeln('Ba so la ba canh tam giac ');
if (a=b) and (b=c) then writeln('- Tam giac deu')
else if (a=b)or(b=c)or(a=c) then writeln('- Tam giac can')
     else if (a*a+b*b=c*c)or(a*a+c*c=b*b)or(b*b+c*c=a*a) then
          writeln('- Tam giac vuong');
p:=(a+b+c)/2; s:= sqrt(p*(p-a)*(p-b)*(p-c));
writeln('Dien tich tam giac la: ',s:1:5);
end
```

```
else writeln('Ba so khong la ba canh tam giac'); readln;
end.
```

Bài 2 chương 2 : giải phương trình bậc hai

```
var a,b,c,x1,x2,d,can: real;
begin write('a,b,c : '); readln(a,b,c); d := b*b - 4*a*c;
if abs(d) < 0.0001 then writeln('Nghiem kep x = ',-b/(2*a):1:5)
else if d<0 then writeln('Phuong trinh vo ngiem')
     else begin can:= sqrt(d); write('x1 = ',(-b+can)/(2*a):1:5);
                write('  x2 = ',(-b-can)/(2*a):1:5); end;
readln; end.
```

Bài 3 chương 2 : giải phương trình trùng phương bậc bốn

```
var a,b,c,d,e,y1,y2,x1,x2,x3,x4: real; i: integer;
begin write('Vao a,b,c = '); readln(a,b,c); d:= b*b-4*a*c;
if d>0 then i:=1 else if d<0 then i:=2 else i:=3;
case i of
1: begin e:= sqrt(d); y1:= (-b+e)/(2*a); y2:= (-b-e)/(2*a);
         writeln('y1 = ',y1:1:5,' y2 = ',y2:1:5);
         if y1>0 then begin x1:= sqrt(y1); x2:= -x1;
                      writeln('x1 = ',x1:1:5,' x2 = ',x2:1:5);end;
         if y2>0 then begin x3:= sqrt(y2); x4:= -x3;
                      writeln('x3 = ',x3:1:5,' x4 = ',x4:1:5); end;
   end;
2: writeln('Phuong trinh vo nghiem');
3: begin y1:= -b/(2*a); writeln('y1 = ',y1:1:5);
         if y1>0 then begin x1:= sqrt(y1); x2:= -x1;
                      writeln('x1 = ',x1:1:5,' x2 = ',x2:1:5); end;

   end;
end;  readln; end.
```

Bài 4 chương 2 : hệ phương trình hai ẩn bậc nhất

```
Var a,b,c,d,e,f,dt,dx,dy: real;
begin write('Vao a,b,c = '); readln(a,b,c);
      write('Vao d,e,f = '); readln(d,e,f);
dt:= a*e-d*b; dx:= c*e-f*b; dy:= a*f-d*c;
if dt=0 then begin if (dx=0) and (dy=0) then
                    writeln('Phuong trinh co vo so nghiem')
                  else writeln('Phuong trinh vo nghiem');
             end
else begin writeln('Co duy nhat nghiem');
           writeln('x = ',dx/dt:1:5,'  y = ',dy/dt:1:5); end;
readln; end.
```

Bài 5 chương 2 : số ngày của một tháng

```
Var songay, thang: byte; nam: integer;
begin write('Cho biet thang thu may (1..12): '); readln(thang);
writeln;
case thang of
4,6,9,11: songay := 30;
2: begin write('Cho biet nam nao: '); readln(nam); writeln;
         if nam mod 4 =0 then songay:=29 else songay := 28;
```

```
   end
else songay:= 31;
end;
writeln('So ngay la = ',songay); readln;
end.
```

Bài 6 chương 2 : đổi chữ số sang tiếng Anh

```
Var n: byte;
begin write('Vao n (1..10) = '); readln(n);
case n of
  1: write('One');  2: write('Two');  3: write('Three');
  4: write('Four'); 5: write('Five'); 6: write('Six');
  7: write('Seven'); 8: write('Eight'); 9: write('Nine'); 10:
write('Ten');
end; readln; end.
```

Bài 7 chương 2 : tám tập hợp các ký tự

```
Var c: char;
begin write('Vao mot ky tu : '); readln(c);
case c of
'0'..'9': writeln('Chu so');
'a'..'z': writeln('Chu thuong');
'A'..'Z': writeln('Chu hoa');
'+','-','*','/': writeln('Cac dau phep tinh');
'>','<','=': writeln('Cac dau so sanh');
'(',')','{','}','[',']': writeln('Cac dau ngoac');
'.',',',';','!': writeln('Cac dau ngat cau');
else writeln('Cac ky tu khac');
end; readln; end.
```

Bài 8 chương 2 : các số nguyên tố nằm giữa hai số

```
label nhan; var i,j,n1,n2: integer;
begin write('n1, n2 = '); readln(n1,n2);
for i:= n1 to n2 do begin for j:=2 to (i div 2) do
                              if i mod j =0 then goto nhan;
                          write(i,'  ');
                          nhan: ;
                    end; writeln; readln;
end.
```

Bài 9 chương 2 : số hoàn hảo

```
Var i,j,n,tong: integer;
begin write('Vao n = '); readln(n);
for j:=1 to n do
  begin tong:= 0;
    for i:=1 to j-1 do if j mod i = 0 then tong:=tong+i;
    if tong=j then begin
        writeln('So hoan hao la : ',j);
        write(' - cac uoc so la : ');
        for i:=1 to j-1 do if j mod i=0 then write(i,' '); writeln;
                  end;
  end;
readln; end.
```

Bài 12 chương 2 : tính hàm e mũ x

```
Var n: integer; emu,eps,x,t: real;
begin write('Vao x, eps = '); readln(x,eps);
emu:= 1; n:= 1; t:= x;
while abs(t) > eps do begin emu:= emu+t; inc(n); t:= t*x/n; end;
writeln('Emu = ',emu:1:5); readln; end.
```

Bài 13 chương 2 : tính hàm sin(x)

```
Var d,p,g,n: integer; x,t,s: real;
Begin write('Vao so do goc d, p, g : '); readln(d,p,g);
x:= (pi/(180.0*3600.0))*(d*3600.0+p*60.0+g);
n:=1; t:=x; s:=x;
repeat t:= -t*x*x/(2*n)/(2*n+1); s:= s+t; inc(n);
until (abs(t)<0.00001);
writeln('Gia tri sin tu tinh la : ',s:1:5);
writeln('Dung ham Sin cua Pascal : ',sin(x):1:5); readln;
End.
```

Bài 14 chương 2 : căn bậc hai của một số

```
Var t,x0,x,r: real;
Begin write('Vao t va x0 = '); readln(t,x0);
repeat x:= (t/x0+x0)/2; r:= x0; x0:= x; until abs(x-r) < 0.0001;
writeln('Can bac hai cua ',t:1:5,' la : ',x0:1:5); readln;
End.
```

Bài 15 chương 2 : tài khoản gửi ngân hàng

```
uses crt;
var sd,gui,rut: real; chon: char;
Begin clrscr; sd:= 0;
repeat writeln;
writeln('X. Xem tai khoan'); writeln('G. Gui tien vao');
writeln('R. Rut tien ra'); writeln('K. Ket thuc');
write('Chon chuc nang : '); readln(chon);
case chon of
'X','x': writeln('So du hien tai la : ',sd:1:2);
'G','g': begin repeat write('Gui vao bao nhieu tien : ');
                      readln(gui);
               until gui>0;
               sd := sd + gui;
               writeln('So du tai khoan moi : ',sd:1:2);
         end;
'R','r': begin repeat write('Rut ra bao nhieu tien : ');
                      readln(rut);
               until sd-rut >= 0;
               sd := sd - rut;
               writeln('So du tai khoan moi : ',sd:1:2);
         end;
end;
until upcase(chon)='K';
End.
```

Bài 1 chương 3 : thao tác cơ bản trên mảng một chiều

```
uses crt;
var a: array[1..40] of integer;
    i,n,sld,j,max,maxtd,tongd,sldluu,sl: integer;
begin clrscr; write('n = '); readln(n);
for i:=1 to n do begin write('a',i,' = '); readln(a[i]) end;
```

{ Cau: so luong so hang duong lien tiep nhieu nhat }

```
j:=1; max:=0;
while j<= n do  if a[j] > 0 then
  begin  sld:=0; i:=j;
    while((i<=n) and (a[i]>0)) do  begin inc(sld); inc(i) end;
    if sld > max then max:=sld;  j:=j+sld;
  end else inc(j);
writeln('So luong so hang duong lien tiep nhieu nhat: ',max);
readln;
```

{ Cau: so luong so hang duong lien tiep co tong lon nhat }

```
j:=1; maxtd:=-1000; sldluu:=0;
while j<= n do if a[j] > 0 then
  begin  sld:=0; i:=j; tongd:=0;
    while((i<=n) and (a[i]>0)) do
       begin inc(sld); tongd:=tongd+a[i];inc(i) end;
    if tongd > maxtd then begin maxtd:=tongd; sldluu:=sld end;
    j:=j+sld;
  end  else inc(j);
writeln('S.luong so hang duong lien tiep co tong lon nhat: ',sldluu);
readln;
```

{ Cau: So luong so hang lien tiep dan dau nhieu nhat }

```
j:=1; max:=0;
while j<= n-1 do if a[j]*a[j+1] < 0 then
  begin  sl:=0; i:=j;
    while((i+1<=n) and (a[i]*a[i+1]<0)) do begin inc(sl); inc(i) end;
    inc(sl); if sl > max then max:=sl; j:=j+sld;
  end  else inc(j);
writeln('So luong so hang  lien tiep dan dau nhieu nhat: ',max);
readln;
end.
```

Bài 2 chương 3 : tần suất của các số trong dãy số

```
var a: array[1..1000] of integer;
    i,j,n,dem,so,t: integer;
begin write('Vao N = '); readln(n);
for i:=1 to n do begin write('a',i,' = '); readln(a[i]) end;
for i:=1 to n-1 do for j:=1 to n-i do if a[j]>a[j+1] then
    begin t:= a[j]; a[j]:= a[j+1]; a[j+1]:= t end;
so:=a[1]; dem:=1;
for i:=2 to n do if a[i]=so then inc(dem)
    else begin writeln('Tan suat cua ',so,' la ',dem);
               so:=a[i]; dem:=1; end;
writeln('Tan suat cua ',so,' la ',dem); readln;
END.
```

Bài 3 chương 3 : tính thu nhập theo lứa tuổi

```
Var n,i,j,k: integer; tuoi: array[1..100] of integer;
    tong: real; tn: array[1..100] of real;
Begin write('So nguoi n = '); readln(n);
for i:=1 to n do
   begin write('Nguoi ',i,'. Nhap tuoi va thu nhap: ');
         readln(tuoi[i],tn[i]); end;
for i:=1 to 100 do
   begin tong:= 0; k:=0;
      for j:=1 to n do if tuoi[j]=i then
           begin tong:= tong+ tn[j]; inc(k) end;
    if k>0 then writeln('Tuoi ',i,': thu nhap t.binh = ',tong/k:1:5);
   end;
readln; end.
```

Bài 5 chương 3 : làm việc với 3 mảng một chiều song song

```
Uses crt;
Var n,i,j: integer; ht: array [1..100] of string[25];
    tk: array[1..100] of longint; sd: array[1..100] of real;
    tong,sd1: real; ht1: string[25]; tk1: longint;
Begin clrscr; write('Vao n = '); readln(n); tong:=0;
for i:=1 to n do begin write('Ho ten: '); readln(ht[i]);
                       write('So tai khoan: '); readln(tk[i]);
                       write('So du: '); readln(sd[i]);
                       tong:= tong+ sd[i];
                 end;
for i:=1 to n-1 do for j:=i+1 to n do if tk[i]>tk[j] then
  begin tk1:= tk[i]; tk[i]:= tk[j]; tk[j]:= tk1;
        ht1:= ht[i]; ht[i]:= ht[j]; ht[j]:= ht1;
        sd1:= sd[i]; sd[i]:= sd[j]; sd[j]:= sd1;
  end;
clrscr;
for i:=1 to n do writeln(ht[i]:25,'  ',tk[i]:9,'  ',sd[i]:1:5);
writeln; writeln('Tong tien du = ',tong:1:5);

readln; end.
```

Bài 8 chương 3 : cộng trừ và nhân hai số dương có hàng ngàn chữ số

```
     { ------------ Cong hai so lon ----------- }
const nmax=2000;
type mang=array[1..nmax] of integer;
var a,b,c: mang; i,n: integer;
begin   clrscr; write('n = '); readln(n);
for i:=1 to n do  begin a[i]:=0; b[i]:=0; c[i]:=0 ; end;
randomize; for i:=1 to n-2 do
              begin a[i]:= random(9); b[i]:=random(9) end;
for i:=n downto 1 do write(a[i]); writeln;
for i:=n downto 1 do write(b[i]); writeln;
for i:=1 to n do c[i]:=a[i]+b[i];
for i:=1 to n do if c[i]>=10 then
     begin c[i+1]:=c[i+1]+ (c[i] div 10);
           c[i]:=c[i] mod 10 end;
for i:=n downto 1 do write(c[i]); writeln;
```

```
readln; end.
```

{ ---------- Tru hai so lon ---------- }

```
uses crt;
const nmax=2000;
type mang=array[1..nmax] of integer;
var a,b,c: mang; i,nho,n: integer;
begin   clrscr; write('n = '); readln(n);
for i:=1 to n do  begin a[i]:=0; b[i]:=0; c[i]:=0 ; end;
randomize;
for i:=1 to n-2 do begin a[i]:= random(9); b[i]:=random(9) end;
a[n-1]:=random(9);
for i:=n downto 1 do write(a[i]); writeln;
for i:=n downto 1 do write(b[i]); writeln;
if a[1]>b[1] then begin c[1]:=a[1]-b[1]; nho:=0 end
   else begin c[1]:= 10+a[1]-b[1]; nho:=1 end;
for i:=2 to n-1 do
   if a[i]>=(b[i]+nho) then begin c[i]:=a[i]-(b[i]+nho); nho:=0 end
        else begin c[i]:=(a[i]+10)-(b[i]+nho);
                   nho:= 1 end;
for i:=n downto 1 do write(c[i]); writeln;
readln; end.
```

{ ------ Tich hai so lon ---------- }

```
uses crt;
const nmax=2000;
type mang=array[1..nmax] of integer;
var a,b,c: mang; i,j,nho,n,n1: integer;
begin   clrscr; write('n = '); readln(n);
for i:=1 to n do  begin a[i]:=0; b[i]:=0; c[i]:=0 ; end;
randomize; n1:=n div 2 -1;
for i:=1 to n1  do begin a[i]:= random(9); b[i]:=random(9) end;
for i:=n1 downto 1 do write(a[i]); writeln; readln;
for i:=n1 downto 1 do write(b[i]); writeln; readln;
for i:= 1 to n1 do for j:= 1 to n1 do c[i+j]:=c[i+j]+a[i]*b[j];
for i:= 1 to n-1 do c[i]:=c[i+1];
for i:= 1 to n do
   begin c[i+1]:=c[i+1]+ (c[i] div 10); c[i]:=c[i] mod 10 end;
for i:=n downto 1 do write(c[i]); writeln;
readln; end.
```

Bài 10 chương 3 : sắp xếp mảng theo chiều xoáy trôn ốc tăng dần

```
uses crt; label 11;
var i,j,k,n: integer;  a: array[1..20,1..20] of integer;
    b: array[1..20*20] of integer; f: text;
Begin  clrscr;assign(f,'dienso.sli'); reset(f); readln(f,n);
for i:=1 to n*n do  read(f,b[i]); writeln; close(f);
for i:=1 to n*n-1 do for j:=1 to n*n-i do if b[j]>b[j+1] then
             begin k:= b[j]; b[j]:=b[j+1]; b[j+1]:= k end;
for i:=1 to n do for j:=1 to n do a[i,j]:=0; k:=1;
for j:=1 to n do begin a[1,j]:=b[k]; inc(k) end;
for i:=2 to n do begin a[i,n]:=b[k]; inc(k) end;
for j:=n-1 downto 1 do begin a[n,j]:=b[k]; inc(k) end;
for i:=n-1 downto 2 do begin a[i,1]:=b[k]; inc(k) end;
```

```
for k:=4*n-3 to n*n do
  begin  if (a[i,j+1]=0)and(a[i-1,j]>0) then
              begin inc(j); a[i,j]:=b[k] ; goto 11 end;
         if (a[i+1,j]=0)and(a[i,j+1]>0) then
              begin inc(i);a[i,j]:=b[k]; goto 11 end;
         if (a[i,j-1]=0)and(a[i+1,j]>0) then
              begin dec(j); a[i,j]:=b[k] ;goto 11 end;
         if (a[i-1,j]=0)and(a[i,j-1]>0) then
              begin dec(i);a[i,j]:=b[k]; goto 11 end;
         11:   ;
  end;
for i:=1 to n do begin for j:=1 to n do write(a[i,j]:4); writeln end;
readln; End.
```

Cấu trúc của tệp dữ liệu: dòng đầu số n (cỡ ma trận a), các dòng dưới là n*n số nguyên viết cách nhau bởi dấu cách.

Bài 12 chương 3 : lập một ma phương

Thuật toán. Đầu tiên ta điền số 1 vào ô ở giữa hàng thứ nhất. Tiếp theo điền các số từ 2 đến n*n cho các ô theo trình tự ưu tiên như sau:

- Nếu ô a[1,n] đã được điền thì ô tiếp theo sẽ điền số là ô a[2,n].
- Nếu ô a[1,j] đã được điền thì ô tiếp theo là ô a[n,j+1] (nhảy xuống dưới).
- Nếu ô a[i,n] đã được điền thì ô tiếp theo là ô a[i-1,1] (về cột đầu).
- Nếu ô a[i,j] đã được điền mà a[i-1,j+1] > 0 (có số rồi) thì ô tiếp theo sẽ là a[i+1,j].
- Nếu không gặp bốn tình huống trên thì khi điền xong ô a[i,j] ta điền tiếp số cho ô a[i-1,j+1].

```
const nmax=19;
var a: array[1..nmax,1..nmax] of integer; i,j,k,s,n: integer;
begin write('Vao co ma tran (n le ) n = '); readln(n);
for i:=1 to n do for j:=1 to n do a[i,j]:=0;
i:=1; j:=(n+1) div 2; a[i,j]:= 1;
for k:=2 to n*n do
begin if (i=1) and (j=n) then inc(i)
  else if i=1 then begin i:=n; inc(j) end
   else if j=n then begin dec(i); j:=1 end
    else if a[i-1,j+1] > 0 then inc(i)
     else begin dec(i); inc(j)  end;  a[i,j]:=k;
end;
for i:=1 to n do begin for j:=1 to n do write(a[i,j]:4); writeln
end;
readln; end.
```

Bài 15 chương 3 : các từ của một xâu chỉ cách nhau một dấu cách

```
var x:string; i:integer;
Begin  write('Go vao mot cau : ');  readln(x);
while x[1]=' ' do delete(x,1,1);
while x[length(x)]=' ' do delete(x,length(x),1); i:=2;
repeat if (x[i]=' ')and(x[i+1]=' ') then delete(x,i,1) else i:=i+1;
until i>length(x); write('Cau chuan la : ',x); readln;
End.
```

Bài 16 chương 3 : trích phần tên của xâu họ và tên

```
var hoten: string[30]; ten: string[7];  k,d: integer;
Begin write('Nhap ho va ten = '); readln(hoten); d:= length(hoten);
while hoten[d]=' ' do dec(d); k:=d;
while hoten[k]<>' ' do dec(k); ten:=copy(hoten,k+1,d-k);
writeln('Phan ten la: ',ten);  readln
END.
```

Bài 18 chương 3 : đếm số từ của một câu

```
var st:string; sotu,d: byte;
Begin writeln('Nhap 1 cau ket thuc bang dau cham'); readln(st);
sotu:= 0; d:= 1;
while st[d]<> '.' do
  if (st[d]<>' ') and ((st[d+1]=' ') or (st[d+1]= '.')) then
       begin inc(sotu); inc(d) end  else inc(d);
writeln('Trong cau co so tu = ',sotu); readln
End.
```

Bài 20 chương 3 : đổi số hệ 10 sang cơ số bất kỳ

```
var n: word; i,b,d,he: byte; c: string[80]; t: string[1];
Begin write(' N he 10 = '); readln(n);
write('Doi sang he nao : '); readln(he); c:='';
repeat  b:= n mod he;  n:= n div he;
  if b >= 10 then t:= chr(b+55) else str(b,t); c:= c+t;
until n=0;  d:= length(c); write('Ket qua : ');
for i:=d downto 1 do write(c[i]); readln
End.
```

Bài 24 chương 3 : cộng trừ nhân hai tập hợp chữ cái hoa

```
uses crt;
type ChuHoa= set of 'A'..'Z';
var A,B,C,D,E: ChuHoa; i,n: integer; ch:char;

procedure NhapTapHop(var X:ChuHoa);
var i: integer;
begin writeln('Vao tap hop, an 0 ket thuc'); X:=[]; i:=0;
repeat inc(i); write('Vao mot ki tu thu ',i:3, ' : ');
  readln(ch); ch:= upcase(ch);
  if (ch >= 'A') and (ch <= 'Z') then X:=X+[ch];
until ch='0';
end;

procedure InTap(X: ChuHoa);
begin n:=0; for ch := 'A' to 'Z' do if ch in X then inc(n);
writeln('So phan tu cua tap la : ',n);
write('Tap hop gom cac phan tu : ');
for ch := 'A' to 'Z' do if ch in X then write(ch:4); writeln;
end;

Begin clrscr; NhapTapHop(A); NhapTapHop(B);
writeln('Tap A : '); InTap(A); writeln('Tap B : '); InTap(B);
C:=A*B; writeln('Tap C = A*B : '); InTap(C);
D:= A+B; writeln('Tap D = A+B : '); InTap(D);
E:=A-B; writeln('Tap E =A-B : '); InTap(E); readln;
```

```
end.
```

Bài 1 chương 4 : hàm USCLN và BSCNN

```
Var a,i,n,u,v: integer;
Function Uscln(x,y: integer): integer;
Begin while x<>y do if x > y then x:=x-y else y:=y-x;
Uscln:= x;
end;
Function Bscnn(x,y: integer): integer;
begin Bscnn := (x*y) div Uscln(x,y); end;
Begin write('N = '); readln(n);
write('Vao so thu nhat : '); readln(a); u:= a; v:= a;
for i:=2 to n do begin write('Vao so thu ',i,' : '); readln(a);
                       u:= Uscln(u,a); v:= Bscnn(v,a);  end;
writeln('Uscln = ',u,'  Bscnn = ',v); readln; end.
```

Bài 2 chương 4 : hàm sắp xếp và hàm tính trung bình của mảng một chiều

```
Type mang= array[1..100] of real;
var a: mang; i,n,k: integer;

function SapXep(a: mang; n: integer): integer;
var tang,giam,bang: byte; i: integer;
begin bang:= 1;
for i:=2 to n do if a[i]<>a[1] then begin bang:=0; break end;
if bang=1 then begin SapXep:= 2;  exit end;
tang := 1;
for i:=1 to n-1 do if a[i]>a[i+1] then
     begin tang:=0; break end;
if tang=1 then begin SapXep:= 1; exit end;
giam := 1;
for i:=1 to n-1 do if a[i]<a[i+1] then
     begin giam:=0; break end;
if giam=1 then begin SapXep:= -1; exit end;
SapXep:= 0;
end;

Begin write('N = '); readln(n);
for i:=1 to n do begin write('a',i,' = '); readln(a[i]) end;
k:= SapXep(a,n);
case k of
1: writeln('Mang da sap xep tang dan');
-1: writeln('Mang da sap xep giam dan');
0: writeln('Mang chua sap xep');
2: writeln('Cac phan tu bang nhau');
end; readln;
end.
```

Lập hàm TrungBinh:

```
Type mang= array[1..100] of real;
var a: mang; i,n,n1,n2: integer;

function TrungBinh(a: mang; n,n1,n2: integer): real;
var tong: real; i: integer;
begin  tong:=0; for i:=n1 to n2 do tong:= tong+ a[i];
TrungBinh:= tong / (n2-n1+1);
```

```
end;
Begin write('N = '); readln(n);
for i:=1 to n do begin write('a',i,' = '); readln(a[i]) end;
write('n1, n2 = '); readln(n1,n2);
writeln('Trung binh = ',TrungBinh(a,n,n1,n2):1:5);
readln;
end.
```

Bài 3 chương 4 : tổng của các giai thừa

```
Var i,n: integer; tong: longint;
function gthua(n: longint): longint;
begin  if (n=0) then gthua:=1  else gthua := n* gthua(n-1); end;
Begin write('N = '); readln(n); tong:=0;
for i:=1 to n do tong:= tong + gthua(i);
writeln('Tong = ',tong); readln;
End.
```

Bài 4 chương 4 : hàm tính tiền lương

```
Var ht: string[25]; ln: real; snc: integer;
Function TienLuong(snc: integer; ln: real): real;
begin if snc>26 then TienLuong:=(26+ (snc-26)*2) * ln
      else TienLuong:= snc * ln;
end;
begin write('Ho ten: '); readln(ht);
write('Luong ngay, so ngay cong: '); readln(ln,snc);
writeln; writeln('Ho va ten: ',ht);
writeln('Tien luong: ',TienLuong(snc,ln):1:5); readln;
end.
```

Bài 5 chương 4 : tính tổng giá trị các mặt hàng

```
const nmax=1000;
type mang=array[1..nmax] of real;
var  s,g: mang; t: real; i,n: integer;
function GiaTri(s,g: mang; n: integer):real;
var j: integer; tg: real;
begin tg:=0; for j:=1 to n do tg:=tg+ s[j]*g[j]; GiaTri:= tg;
end;
Begin write('n = '); readln(n);
for i:=1 to n do begin write('s[',i,'] = '); readln(s[i]) end;
for i:=1 to n do begin write('g[',i,'] = '); readln(g[i]) end;
t:= GiaTri(s,g,n);
writeln('Tong gia tri cac mat hang la : ',t:1:5); readln;
End.
```

Bài 6 chương 4 : tính giá trị của một đa thức

```
const nmax=20;
type mang=array[0..nmax] of real;
var a: mang; i,n: integer; x: real;
function DaThuc(a: mang; n: integer; x: real): real;
var i: integer; tong,r: real;
begin tong:= a[0]; r:= x;
for i:=1 to n do begin tong:= tong + a[i]*r; r:= r*x end;
```

```
Dathuc:= tong;
end;
Begin write('Bac da thuc n = '); readln(n);
for i:=n downto 0 do begin write('a[',i,'] = '); readln(a[i]) end;
write('x = '); readln(x);
writeln('Gia tri cua da thuc = ',DaThuc(a,n,x):1:5); readln
End.
```

Bài 7 chương 4 : tìm nghiệm của phương trình phi tuyến f(x) = 0

```
var a,b,c:real;
function  f(x:real):real; begin f:=x*x-2; end;
begin  write('Vao hai so a,b:  '); readln(a,b); {nhap 0 va 10}
if f(a)*f(b)>0 then  writeln('Khong thoa dieu kien dau bai')
else begin  while (f(a)<>0) and (f(b)<>0) and (b-a>=0.00001) do
              if f((a+b)/2)*f(a)<0 then b:=(a+b)/2 else a:=(a+b)/2;
            if f(b)=0 then writeln('Nghiem la : ',b:1:5)
                      else writeln('Nghiem la : ',a:1:5)
     end; readln;
end.
```

Bài 8 chương 4 : phân tích một số thành tích các số nguyên tố

```
var n: longint;
procedure tsnto(n:longint);
label 11;  var i,n1: longint;
begin write(n,' = ');
11: n1:= n div 2;
for i:=2 to n1 do  if n mod i = 0 then
     begin write(i,' . ');  n:= n div i;  goto 11;  end;
writeln(n);
end;
begin write('Vao n = '); readln(n); tsnto(n); readln
end.
```

Bài 9 chương 4 : nối hai mảng làm một và sắp xếp cả ba mảng

```
const nmax=100;
type mang= array[1..nmax] of real;
var a,b,c: mang;  m,k,i: integer;
procedure XepTang(n: integer; var s: mang);
var tgian:real; i,j: integer;
begin  for i:=1 to n-1 do for j:=i+1 to n do  if s[i]>s[j] then
          begin tgian:=s[i]; s[i]:=s[j]; s[j]:=tgian; end;
end;
begin write('M = ');readln(m);
for i:=1 to m do begin write('a[',i,'] = '); readln(a[i]); end;
write('K = '); readln(k);
for i:=1 to k do begin write('b[',i,'] = '); readln(b[i]); end;
for i:= 1 to m do c[i]:=a[i];
for i:=1 to k do c[m+i]:=b[i];
XepTang(m,a); XepTang(k,b); XepTang(m+k,c);
writeln('Mang a da xep:');
for i:=1 to m do
   begin write(a[i]:13:5); if i mod 5=0 then writeln; end;
writeln; writeln('Mang b da xep:');
```

```
for i:=1 to k do
   begin write(b[i]:13:5);if i mod 5=0 then writeln; end;
writeln; writeln('Mang c da xep:');
for i:=1 to m+k do
   begin write(c[i]:13:5); if i mod 5=0 then writeln; end;
readln;
end.
```

Bài 10 chương 4 : giải bài toán tháp Hà nội dùng đệ quy

```
var n: byte; d: integer;
procedure chuyen(n,c1,c2,c3: byte);
begin if n=1 then
         begin inc(d); write('Lan thu ',d,' : ');
            writeln(' chuyen dia tu coc ',c1,' sang coc ',c2) end
      else  begin chuyen(n-1,c1,c3,c2); chuyen(1,c1,c2,c3);
                  chuyen(n-1,c3,c2,c1) end
end;
begin write('Vao so dia n = '); readln(n); d:=0;
writeln('So dia can chuyen n = ',n); chuyen(n,1,2,3); readln;
end.
```

Bài 2 chương 5 : bài toán tuyển sinh

```
uses crt;
const nmax= 100;
type date = record ngay,thang,nam: word; end;
     thisinh = record sbd: string[10];
                      hoten: string[25];
                      ngaysinh : date;
                      mon1,mon2,mon3,uutien,tongso: real;
               end;
     mangts = array[1..nmax] of thisinh;
var a: mangts; chon,n: integer;

procedure Nhap;
var ans: integer;
begin while true do begin inc(n);
with a[n],ngaysinh do
  begin
   write('Vao so bao danh : '); readln(sbd);
   write('Vao ho ten thi sinh : '); readln(hoten);
   write('Vao ngay, thang, nam sinh : '); readln(ngay,thang,nam);
   write('Diem mon1, mon2, mon3, uutien : ');
   readln(mon1, mon2, mon3, uutien);
   tongso:= mon1 + mon2 + mon3 + uutien;
  end;
write('Co tiep tuc nhap khong 1/0 ? '); readln(ans);
if ans=0 then break;
                    end;
end;

procedure Duyet;
var i: integer;
begin  for i:=1 to n do with a[i],ngaysinh do
writeln(sbd:10,' ',hoten:25,' ',ngay:2,'/',thang:2,'/',nam:4,' ',
mon1:5:2,' ',mon2:5:2,' ',mon3:5:2,' ',uutien:1:0,' ',tongso:5:2);
```

```
readln;
end;

procedure SXepSbd;
var i,j: integer; x: thisinh;
begin for i:=1 to n-1 do for j:= i+1 to n do
      if a[i].sbd > a[j].sbd then
         begin x:= a[i]; a[i]:= a[j]; a[j]:= x end;
end;

procedure TraCuuDiem;
var mid,low,high,k: integer; x: string[10];
begin SXepSbd; write('Vao so bao danh can tra diem: '); readln(x);
low:=1; high:=n; k:=0;
while low<=high do
  begin mid:= (low+high) div 2;
        if x < a[mid].sbd then high:= mid-1
        else if x > a[mid].sbd then low:= mid+1
             else begin k:=mid; break end;
  end;
if k<>0 then with a[k],ngaysinh do
   writeln(sbd:10,' ',hoten:25,' ',mon1:5:2,' ',mon2:5:2,
                   ' ',mon3:5:2,' ',uutien:1:0,' ',tongso:5:2)
else writeln('Khong co so bao dang nay');
readln;
end;

procedure TrungTuyen;
var dchuan: real; i: integer;
begin SXepSbd; write('Vao diem chuan : '); readln(dchuan);
writeln('Danh sach thi sinh trung tuyen : ');
for i:=1 to n do  with a[i],ngaysinh do
  if (tongso>=dchuan) and (mon1>1) and (mon2>1) and (mon3>1) then
  writeln(sbd:10,' ',hoten:25,' ',ngay:2,'/',thang:2,'/',nam:4,' ',
  mon1:5:2,' ',mon2:5:2,' ',mon3:5:2,' ',uutien:1:0,' ',tongso:5:2);
readln;
writeln('Danh sach thi sinh khong trung tuyen : ');
for i:=1 to n do  with a[i],ngaysinh do
  if (tongso<dchuan) or (mon1<=1) or (mon2<=1) or (mon3<=1) then
  writeln(sbd:10,' ',hoten:25,' ',ngay:2,'/',thang:2,'/',nam:4,' ',
  mon1:5:2,' ',mon2:5:2,' ',mon3:5:2,' ',uutien:1:0,' ',tongso:5:2);
readln;
end;

procedure DsDiemGiam;
var i,j: integer; x: thisinh;
begin for i:=1 to n-1 do for j:= i+1 to n do
      if a[i].tongso < a[j].tongso then
         begin x:= a[i]; a[i]:= a[j]; a[j]:= x end;
for i:=1 to n do with a[i],ngaysinh do
     writeln(sbd:10,' ',hoten:25,' ',tongso:5:2);
readln;
end;

Begin  clrscr; n:=0;
repeat
writeln('1. Nhap du lieu cho cac thi sinh');
```

```
writeln('2. Duyet danh sach');
writeln('3. Sap xep theo so bao danh tang dan va tra cuu diem');
writeln('4. In danh sach trung tuyen va khong trung tuyen');
writeln('5. In danh sach theo tong so diem giam dan');
writeln('6. Ket thuc chuong trinh');
write('Chon chuc nang : '); readln(chon);
case chon of
1: Nhap;
2: Duyet;
3: TraCuuDiem;
4: TrungTuyen;
5: DsDiemGiam
end;
until chon = 6;
End.
```

Chức năng 1 cho phép nhập bổ sung các thí sinh mới.

Bài 2 chương 6 : bán vé máy bay

```
uses crt;
const nmax = 30;
const tp: array[1..7] of string[12]=('Vinh','Hue','Danang',
            'Nhatrang','Dalat','Saigon','Cantho');
      tien: array[1..7] of real = (500000,1000000,1200000,1700000,
             1800000,2100000,2300000);
type chuyenbay = record dich: string[12];
                        ngay: string[10];
                        gio: string[5];
                        cho: array[1..100] of byte;
                 end;
     mangchuyenbay = array[1..nmax] of chuyenbay;
     tepchuyenbay = file of chuyenbay;
var a: mangchuyenbay; f: tepchuyenbay; chon: integer;

procedure Nhap;
var x: chuyenbay; i: integer;
begin assign(f,'CBay.dat'); {$I-} reset(f);  {$I+}
if IOResult <> 0 then rewrite(f) else seek(f, filesize(f));
with x do
  begin write('Vao dich chuyen bay : '); readln(dich);
        write('Vao ngay bay DD/MM/YYYY : '); readln(ngay);
        write('Vao gio bay HH.MM : '); readln(gio);
        for i:=1 to 100 do cho[i]:= 0;
  end;
write(f,x); close(f);
end;

procedure Duyet;
var x: chuyenbay; i: integer;
begin assign(f,'CBay.dat'); {$I-} reset(f); {$I+}
if IOResult<>0 then begin writeln('Khong ton tai tep'); exit end;
if filesize(f)=0 then
         begin writeln('Khong co ban ghi nao'); exit end;
while not eof(f) do
  begin  read(f,x);
```

```
    with x do write(dich:12,' ',ngay:10,' ',gio:5,' ');
    for i:=1 to 40 do if i mod 10 =0 then write(x.cho[i],' ')
                     else write(x.cho[i]); writeln;

  end;
close(f);
end;

procedure BanVe;
label L1;
var ht: string[25]; cb: string[12]; n: string[10]; g: string[5];
    c,ans,i,dem: integer; x: chuyenbay; found: boolean;
begin Duyet; write('Co dat cho khong 1/0 ?  '); readln(ans);
if ans=0 then exit;
write('Vao ten hanh khach : '); readln(ht);
write('Vao dich chuyen bay : '); readln(cb);
write('Vao ngay muon bay DD/MM/YYYY : '); readln(n);
write('Vao gio muon bay HH.MM : '); readln(g);
write('Vao so ghe ngoi khach muon 1..100: '); readln(c);
assign(f,'CBay.dat'); reset(f);  found:= false; dem:=0;
while not eof(f) do
  begin read(f,x); inc(dem);
       if (x.dich=cb) and (x.ngay=n) and (x.gio=g) then Begin
       found:= true;
       L1: if x.cho[c]=0 then
        begin  writeln('---------- IN VE MAY BAY ------------');
               writeln('Ho va ten khach hang : ',ht);
               writeln('Chuyen bay tu Ha noi di ',cb);
               writeln('Ngay bay : ',n);
               writeln('Gio bay : ',g);
               writeln('So ghe ngoi : ',c);
               for i:=1 to 7 do if tp[i]=cb then
               begin writeln('Gia tien : ',tien[i]:10:0); break end;
               readln;
               x.cho[c]:= 1; break;
        end
       else begin  writeln('Cho da co nguoi dat !');
               for i:=1 to 100 do
                     if i mod 10 =0 then write(x.cho[i],' ')
                     else write(x.cho[i]); writeln;
               write('Co dat cho khac khong 1/0 ? '); readln(ans);
               if ans=0 then break;
               write('Vao cho khac 1..100 : '); readln(c);
               goto L1;
            end;                                              End;

  end;
if not found then writeln('Khong co chuyen bay nay')
             else begin reset(f); seek(f,dem-1); write(f,x) end;
close(f);
end;

procedure Xoa;
var i,j,n: integer; ngay1,ngay2,x: string[10];
    gio1: string[5]; ch: char;
begin assign(f,'CBay.dat'); reset(f); n:= filesize(f);
for i:=1 to n do read(f,a[i]); close(f);
```

```
write('Vao ngay hien tai YYYY/MM/DD : '); readln(ngay1);
write('Vao thoi gian hien tai HH.MM : '); readln(gio1);
assign(f,'CBay.dat'); rewrite(f);
for i:=1 to n do
 begin  x:= a[i].ngay; ngay2:='';
    ngay2:=ngay2+ copy(x,7,4)+'/'+copy(x,4,2)+'/'+copy(x,1,2);
    writeln('ngay2 = ',ngay2:10,' a[',i,'].gio = ',a[i].gio:5);
    if (ngay2>ngay1) or ((ngay2=ngay1)and(a[i].gio>gio1)) then
        begin write(f,a[i]); write('da ghi'); readln; end;
 end;
close(f);
end;

Begin clrscr;
repeat  writeln;
writeln('1. Nhap mot chuyen bay moi vao tep Ve.dat');
writeln('2. In CSDL Ve.dat');
writeln('3. Dat cho va in ve cho khach hang');
writeln('4. Xoa tat ca cac chuyen bay da bay');
writeln('5. Ket thuc chuong trinh');
writeln; write('Hay chon chuc nang : '); readln(chon);
case chon of
1: Nhap;    2: Duyet;    3: BanVe;    4: Xoa;
end;
until chon=5;
End.
```

Mảng TP lưu đích các chuyến bay từ Hà nội đi các tỉnh, mảng TIEN lưu giá vé ứng với từng chuyến bay. Chương trình điều khiển theo menu. Thủ tục Nhap nhằm nhập một chuyến bay mới, với mỗi chuyến bay cần nhập tên chuyến bay (đích của chuyến bay phải là một trong các phần tử của mảng TP, chữ cái đầu tiên viết hoa), ngày bay nhập theo định dạng dd/mm/yyyy (hai ký tự cho ngày, vạch chéo, hai ký tự cho tháng, gạch chéo, bốn ký tự cho năm), giờ cất cánh theo định dạng hh.mm (2 ký tự cho giờ, dấu chấm, 2 ký tự cho phút), mảng CHO máy tự điền tất cả các phần tử bằng 0. Mẫu nhập dữ liệu cho một số chuyến bay để đưa vào tệp CBay.dat:

```
Hue     09/10/2003   12.30
Saigon  10/10/2003   09.45
Dalat   10/10/2003   10.50
Cantho  11/10/2003   15.45
```

Thủ tục Duyet: duyệt tất cả các chuyến bay trong tệp CBay.dat, mảng CHO chỉ in 40 chỗ đầu tiên theo 4 cụm số.

Thủ tục BanVe: bán vé cho khách. Khi một người đến mua vé cần nhập họ tên khách, tên chuyến bay (tên chuyến bay phải trùng với một tên trong mảng TP, ví dụ Saigon), nhập ngày mà khách muốn bay theo định dạng dd/mm/yyyy (ví dụ 10/10/2003), nhập giờ bay theo định dạng hh.mm (ví dụ 09.45), nhập số ghế mà khách muốn (ví dụ: ghế số 3). Tiếp theo máy tìm trong cơ sở dữ liệu chuyến bay này, nếu chỗ còn trống thì in vé cho khách và đánh dấu chỗ này đã có người mua rồi. Nếu chỗ đã có người mua thì máy in ra tình trạng của toàn bộ 100 chỗ ngồi, nếu còn chỗ trống thì nhập vào một chỗ khác để bán vé, nếu cả 100 chỗ đã bán hết thì kết thúc thủ tục, chuyến bay đã bán hết vé.

Thủ tục Xoa: xoá khỏi CSDL các chuyến bay đã bay. Chương trình đòi hỏi nhập vào ngày hiện tại theo định dạng yyyy/mm/dd (4 ký tự đầu chỉ năm, dấu gạch chéo, 2 ký tự chỉ tháng, gạch chéo, 2 ký tự chỉ ngày, ví dụ 2003/10/10), nhập thời gian hiện tại theo định dạng hh.mm (ví dụ 10.00). Máy chuyển các bản ghi trong CSDL ra mảng a, sau đó ghi lại vào tệp chỉ những chuyến bay còn chưa thực hiện.

Bài 2 chương 7 : lập một menu dọc trong chế độ văn bản

```
uses crt;
type  str40=string[40];
      bangkt=array[1..20] of str40;
var nd:bangkt; chon:integer;

procedure hienkt(dkt:str40;h,c,dai,mn,mc:integer);
begin textbackground(mn); textcolor(mc); window(c,h,c+dai-1,h);
clrscr; if length(dkt) >= dai then dkt[0]:=chr(dai-1);
write(dkt); gotoxy(1,1);
end;

procedure menu(n,hd,cd,dai:integer;nd:bangkt;
               mnc,mcc,mnr,mcr:integer;var chon:integer);
var i,chonm,mn,mc:integer; ch1,ch2:char;
Begin  if (n<=0) or (n>20) then exit;
if (chon<0) or (chon>n) then chon:=1; chonm:=chon;
for i:=1 to n do hienkt(nd[i],hd+i-1,cd,dai,mnc,mcc);
hienkt(nd[chonm],hd+chonm-1,cd,dai,mnr,mcr); ch1:=chr(0);
while ord(ch1)<>13 do
 begin ch1:=readkey; if ord(ch1)=0 then ch2:=readkey;
  if ord(ch1)=0 then if ord(ch2)=72 then
    begin chonm:=chon-1; if chonm<=0 then chonm:=n; end
  else if ord(ch2)=80 then
         begin  chonm:=chon+1; if chonm>n then chonm:=1; end;
  if chonm<>chon then
    begin hienkt(nd[chon],hd+chon-1,cd,dai,mnc,mcc);
        hienkt(nd[chonm],hd+chonm-1,cd,dai,mnr,mcr); chon:=chonm;
    end
 end;
End;

procedure chunhat; begin write('Chu nhat')  end;
procedure hinhthang; begin write('Hinh thang') end;
procedure duongtron; begin write('Duong tron') end;
procedure hinhcau; var r: real;
begin writeln('HINH CAU'); write('Ban kinh : '); readln(r);
writeln; writeln('The tich : ',4*pi*r*r*r/3:1:5);
end;

BEGIN textmode(C80); textbackground(Blue);
window(1,1,80,25); clrscr;
nd[1]:=' 1. Tinh dien tich hinh chu nhat';
nd[2]:=' 2. Tinh dien tich hinh thang   ';
nd[3]:=' 3. Tinh chu vi duong tron      ';
nd[4]:=' 4. Tinh the tich hinh cau      ';
nd[5]:=' 5. Ket thuc chuong trinh       '; chon:=1;
textbackground(LightGray);
textcolor(Red); window(23,3,55,6);
```

```
clrscr; writeln('  TINH TOAN HINH HOC');
writeln('     Lap trinh: Bui The Tam');
writeln('     Ngay 20 thang 12 nam 1985');
while chon<>5 do
  begin menu(5,8,23,33,nd,cyan,yellow,brown,white,chon);
    textbackground(LightGray); textcolor(LightCyan);
    window(23,15,55,21); clrscr; gotoxy(3,1);
    if chon=1 then  chunhat;
    if chon=2 then  hinhthang;
    if chon=3 then  duongtron;
    if chon=4 then  hinhcau;
  end; textbackground(Black); textcolor(white); window(1,1,80,25);
clrscr; END.
```

Bài 5 chương 8: danh sách số nguyên dùng mảng

```
uses crt;
const max=100;
type danhsach = record a: array[1..max] of integer;
                       last : integer;
                end;
var ds: danhsach; x,p,chon: integer;

Procedure KhoiTao(var ds: danhsach);
Begin ds.last := 0; end;

Procedure InDS(ds: danhsach);
var j: integer;
Begin if ds.last<>0 then
         begin for j:=1 to ds.last do write(ds.a[j],'  ');
               writeln;
         end else writeln('Danh sach rong');
End;

Procedure Nhap(x: integer; var ds: danhsach);
Begin with ds do begin inc(last); a[last]:= x end;
end;

Procedure ChenP(x,p: integer; var ds: danhsach);
var j: integer;
begin  with ds do
  begin for j:= last downto p do a[j+1]:= a[j]; a[p]:= x; inc(last);
  end;
end;

Procedure XoaP(p: integer; var ds: danhsach);
var j: integer;
begin  with ds do
   begin for j:= p to last-1 do a[j]:= a[j+1]; dec(last) end;
end;

Procedure XoaX(x: integer; var ds: danhsach);
var i,j: integer;
Begin with ds do
 begin i:=1;
   while i<= last -1 do
```

```
     begin if a[i]=x then
            begin for j:=i to last-1 do a[j]:=a[j+1]; dec(last) end;
           inc(i);
     end;
   if a[last]=x then dec(last);
 end;
End;

Procedure XoaLap(var ds: danhsach);
var i,j,k: integer;
begin  with ds do
 begin  k:= 1;
   while k<=last-1 do
     begin j:= k+1;
           while j<= last-1 do
             begin if a[j]=a[k] then
                     begin for i:=j to last-1 do a[i]:= a[i+1];
                           dec(last);
                     end;
                   inc(j);
             end;
           if a[last] = a[k] then dec(last);
           inc(k);
     end;
 end;
end;

Begin clrscr;
repeat writeln('1. Tao DS rong   2. Nhap phan tu  3. In DS');
       writeln('4. Chen vi tri p 5. Xoa vi tri p  6. Xoa x');
   write('7. Xoa lap       8. Ket thuc      Chon : '); readln(chon);
case chon of
1: KhoiTao(ds);
2: Begin write('Vao x = '); readln(x); Nhap(x,ds); End;
3: InDS(ds);
4: Begin write('Vao x, vi tri chen p = ');
         readln(x,p); ChenP(x,p,ds); End;
5: Begin write('Vi tri p can xoa : '); readln(p); XoaP(p,ds); End;
6: Begin write('Vao x = '); readln(x); Xoax(x,ds); End;
7: XoaLap(ds);
end; writeln;
until chon=8;
End.
```

Bài 6 chương 8: danh sách số nguyên dùng danh sách liên kết

```
uses crt;
type  NodePtr = ^Node;
      Node = record so: integer;
                    next : NodePtr;
             end;
var list,last: NodePtr; x,chon: integer;

procedure KhoiTao; begin list := NIL; last := NIL; end;

procedure Nhap(x: integer);
var p: NodePtr;
```

```
begin new(p); p^.so:=x; p^.next:=NIL;
if list=NIL then begin list:= p; last:=p end
            else begin last^.next := p; last := p end;
end;

Procedure InDS;
var p: NodePtr;
Begin if list<>NIL then
         begin p:= list;
           while p<>NIL do begin write(p^.so,'  '); p:=p^.next end;
           writeln;
         end else writeln('Danh sach rong');
End;
```

{ Chen mot phan tu moi vao sau mot phan tu co gia tri x }

```
Procedure ChenSauX(x: integer);
var p,moi: NodePtr; found: boolean;
begin  if list=NIL then begin writeln('DS rong'); exit end;
p:= list; found:= false;
while (p<>NIL) and (not found) do
   if p^.so=x then found:= true else p:=p^.next;
if found then begin new(moi);
   write('Vao phan tu moi: '); readln(moi^.so);
   moi^.next:= p^.next; p^.next:= moi;
              end else writeln('Khong tim thay');
end;

Procedure XoaLap;
var p,q,r: nodePtr;
begin p:= list;
while p<>NIL do
 begin q:= p; r:= q^.next;
    while r<>NIL do
      begin if r^.so=p^.so then begin q^.next:= r^.next;
                                      r:= r^.next
                                end
            else begin q:=r;r:= r^.next end;
      end;
    p:= p^.next;
 end;
end;
Begin clrscr;

repeat writeln('1. Tao DS rong   2. Nhap phan tu  3. In DS');
       writeln('4. Chen sau x    5. Chen truoc x  6. Xoa sau x');
   write('7. Xoa x  8. Xoa lap  9. Ket thuc.  Chon : ');
readln(chon);
case chon of
1: KhoiTao;
2: Begin write('Vao x = '); readln(x); Nhap(x); End;
3: InDS;
4: Begin write('Vao x = '); readln(x); ChenSauX(x); end;
8: XoaLap;
end; writeln;
until chon=9;
End.
```

Bài 7 chương 8: cộng hai đa thức dùng mảng

```
const max=50;
type sohang = record heso: real; bac: integer end;
     dathuc = record pt : array[1..max] of sohang;
                     n: integer;
              end;
var P,Q,T : dathuc;

procedure Nhap(var P: dathuc);
var i: integer;
begin P.n:= 0;
for i:=1 to max do begin P.pt[i].heso:=0; P.pt[i].bac:=0 end;
with P do
  begin write('Vao n = '); readln(n);
        for i:=1 to n do begin write('So hang thu ',i,' co ');
                               write('he so va bac = ');
                               readln(pt[i].heso,pt[i].bac);
                         end;
  end;
end;

Procedure Print(P: dathuc);
var i : integer;
begin  write('Da thuc : ');
with P do
  begin for i:=1 to n do
          write('(',pt[i].heso:1:2,', ',pt[i].bac,')  ');
        writeln;
  end;
end;

Procedure Cong(P,Q: dathuc; var T: dathuc);
var i,j: integer; x: sohang;
begin   { noi da thuc Q vao sau da thuc P }
with P do
  begin for i:=1 to n do begin T.pt[i].heso:= pt[i].heso;
                               T.pt[i].bac := pt[i].bac; end;
        T.n:= n;
  end;
j:= T.n;
with Q do
  begin for i:=1 to n do begin T.pt[j+i].heso:= pt[i].heso;
                               T.pt[j+i].bac := pt[i].bac; end;
        T.n:= j+n;
  end;
{ Sap xep T theo thuat toan noi bot }
with T do
  begin for i:=1 to n-1 do for j:=1 to n-i do
        if pt[j].bac < pt[j+1].bac then
          begin x:= pt[j]; pt[j]:= pt[j+1]; pt[j+1]:= x end;
  end;
{ Cong cac so hang cung bac trong T, neu he so bang 0 thi xoa }
with T do
 begin i:=1;
    while i<= n-1 do
```

```
      begin if pt[i].bac = pt[i+1].bac then
              begin pt[i].heso:= pt[i].heso + pt[i+1].heso;
                    for j:=i+1 to n-1 do pt[j]:= pt[j+1];
                    dec(n);
              end;
            inc(i);
      end;
    for i:=1 to n do if pt[i].heso=0 then
      begin for j:=i to n-1 do pt[j]:= pt[j+1];
            dec(n);
      end;
 end;
end;
Begin Nhap(P); Print(P); readln; Nhap(Q); Print(Q); readln;
Cong(P,Q,T); Print(T);
readln; End.
```

Chương trình trên làm nhiệm vụ: nhập hai đa thức p và q, cộng hai đa thức p và q để được đa thức kết quả là t. Ví dụ các số hạng (hệ số và bậc) của các đa thức:

p: (11,11) (9,9) (6,5) (2,2) (1,1) (10,0)
q: (7,7) (-6,5) (3,3) (1,2)
t: (11,11) (9,9) (7,7) (3,3) (3,2) (1,1) (10,0)

Bài 8 chương 8 : cộng hai đa thức dùng danh sách liên kết

```
uses crt;
type Ptr = ^sohang;
     Sohang = Record
                 Heso: real;
                 Bac : integer;
                 Next: Ptr;
               End;
Var P,Q,T: Ptr;

Procedure TaoRong(var P: Ptr);
Begin P:= nil end;

Procedure Nhap(var P: Ptr);
var R,last: ptr; tt: integer;
Begin  Taorong(P);
repeat new(R); write('He so - Bac : '); readln(R^.heso,R^.bac);
       R^.next:= nil;
       if p=nil then begin P:=R; last:= R end
                else begin last^.next:=R; last:=R end;
       write('tiep tuc nhap 1/0 ? '); readln(tt);
until tt=0;
End;

procedure Print(P: ptr);
var C: ptr;
begin  if c=nil then begin writeln('Rong'); exit end; C:=P;
while C<> nil do begin write(C^.heso:7:2,'  ',C^.bac,' // ');
                       C:=C^.next
                 end; writeln;
end;

Procedure Cong(P,Q: ptr; var T: Ptr);
```

```
Label L1,L2,xongR;
Var R,q1,q2: Ptr;
begin  T:= Q;
L1: if P^.bac > T^.bac then
        begin R:= P; P:= P^.next; R^.next:= T; T:= R end
else  begin if P^.bac = T^.bac then
               Begin T^.heso:= T^.heso + P^.heso; P:= P^.next;
                     if T^.heso=0 then begin T:=T^.next; goto L1 end
                                  else goto 12;
               End;
      end;  { Truong hop P^.bac < T^.bac }
L2: while (P<> nil)  do
  Begin  R:= P;  P:= P^.next; Q1:= T; Q2:= T^.next;
     while q2<>nil do
      Begin
        if R^.bac > Q2^.bac then
             begin  Q1^.next := R; R^.next := Q2; goto xongR; end;
        if R^.bac = Q2^.bac then
             begin Q2^.heso := Q2^.heso + R^.heso;
                   if Q2^.heso = 0 then  Q1^.next:= Q2^.next;
                   goto xongR;
             end;
        if R^.bac < Q2^.bac then begin  q1:= q2; q2:= q1^.next end;
      End;
    if (q2=nil) and (q1<>nil) then
      begin if (r^.bac=q1^.bac) then
              begin q1^.heso:= q1^.heso + r^.heso;
                    if q1^.heso=0 then begin q1^.next:=NIL;
                                             goto xongR;
                                       end
                                  else q1^.next := NIL;
              end
            else begin q1^.next:=R; R^.next:=nil end;
      end;
    xongR:
  End;
end;

Begin  clrscr; Nhap(P); write('Da thuc P: ');Print(P); readln;
Nhap(Q); write('Da thuc Q: ');Print(Q); readln;
Cong(P,Q,T);write('Da thuc T: '); Print(T); readln;
End.
```

Ví dụ 1: các số hạng (hệ số và bậc) của các đa thức

p: (11, 11) (9, 9) (6, 5) (2, 2) (1, 1) (10, 0)

q: (7, 7) (-6, 5) (3, 3) (1, 2)

t: (11, 11) (9, 9) (7, 7) (3, 3) (3, 2) (1, 1) (10, 0)

Ví dụ 2: p : (-3, 3) (4, 2) (-6 1) (2, 0)

q : (3, 3) (-4 , 2) (7, 1)

t : (1, 1) (2, 0)

Bài 9 chương 8 : danh sách liên kết thuận luôn luôn được sắp xếp

```
type Ptr = ^Node;
     Node = record so: integer; next: Ptr end;
```

```
var list:^Ptr; x,chon: integer;

procedure Them(x: integer);
var moi,q,p: Ptr;
begin new(moi); moi^.so := x;
if list=nil then begin moi^.next:= nil; list:= moi; exit end;
q:= nil; p:= list;
if x <= p^.so  then begin moi^.next:= p; list:= moi; exit end;
while p<>nil do begin if p^.so >= x then break;
                      q:= p; p:= p^.next
                end;
if p=nil then begin moi^.next:= nil; q^.next:= moi; exit end
         else begin moi^.next:= p; q^.next:= moi end;
end;

procedure Duyet;
var p: Ptr;
begin  p:= list;
while p<>nil do begin write(p^.so,'  '); p:= p^.next end; writeln;
end;

Begin  list := nil;
repeat write('1. Them  2. Duyet  3. Ket thuc. Chon : ');
readln(chon);
case chon of
1: begin write('Vao x = '); readln(x); Them(x) end;
2: Duyet;
end;
until chon = 3;
End.
```

Bài 11 chương 8 : tính giá trị biểu thức dùng biểu thức Balan

Cách nhập một biểu thức số học: mỗi số thực hay nguyên, mỗi dấu phép toán, mỗi dấu mở ngoặc hay đóng ngoặc đều nhập trên một dòng, kết thúc nhập là một dòng chứa một ký tự là dấu #. Ví dụ nhập vào biểu thức 2.156 * (9.17 + 6.23) - 15.75 / 5.45 # sẽ cho kết quả 30.317781. Hàm Pri() cho độ ưu tiên trên các ký tự S(+-*/

```
uses crt;
type string10=string[10];
var e,e1,s:array[1..20] of string[10]; i,j,n,top: integer;
    x,y,r: real; z,u,v,w: string10;

function pri(z: string10): byte;
begin pri:=0; if z='$' then pri:=1; if z='(' then pri:=2;
if (z='+')or(z='-') then pri:=3; if (z='*') or (z='/') then pri:= 4;
end;

procedure BTBalan;
begin  s[1]:='$'; top:=1; j:=0; z:=e[1]; i:=1;
while z <> '#' do
 begin if z='(' then begin inc(top); s[top]:=z end
       else  if z=')' then begin  while s[top]<> '(' do
                                    begin inc(j); e1[j]:=s[top];
                                          s[top]:=''; dec(top);
                                    end;
                                  s[top]:=''; dec(top);
```

```
                              end
   else if (z='+') or (z='-') or (z='*') or (z='/') then
           begin while Pri(s[top]) >= pri(z) do
                     begin inc(j); el[j]:=s[top];
                           s[top]:=''; dec(top);
                     end;
                inc(top); s[top]:=z;
           end  else begin inc(j); el[j]:=z end;
 inc(i); z:=e[i];
 end;
while s[top]<>'$' do
  begin inc(j); el[j]:=s[top]; s[top]:=''; dec(top); end;
inc(j); el[j]:='#';
end;

function GiaTri: real;
begin j:=1; z:=El[j]; top:=0;
while z<>'#' do
 begin
   if (z='+')or(z='-')or(z='*')or(z='/')or(z='(')or(z=')') then
     begin  v:= S[top]; dec(top); u:= s[top]; dec(top);
            val(u,x,i); val(v,y,i);
            if z='+' then r:=x+y; if z='-' then r:=x-y;
            if z='*' then r:=x*y; if z='/' then r:=x/y;
            str(r:10:5,w); inc(top); s[top]:=w;
     end else begin inc(top); s[top]:=z end;
   inc(j); z:= el[j];
 end;
val(s[top],x,i); GiaTri:=x;
end;

Begin  clrscr;
writeln('Hay nhap cac so, + - * / ( ) # '); i:=1;
repeat  readln(z); E[i]:=z; inc(i);  until z='#';
writeln('Bieu thuc toan hoc : '); n := i-1;
for i:= 1 to n do write(e[i],'  '); writeln; writeln;
BTBalan;
writeln('Bieu thuc Ba lan: '); j:=1;
repeat  write(el[j],'  '); z:=el[j]; inc(j);
until z='#'; writeln; writeln;
writeln('Gia tri cua bieu thuc la : ',GiaTri:13:5);
readln;
End.
```

Bài 1 chương 9 : đọc in xâu ký tự trên màn hình đồ hoạ

```
uses graph,crt;
type str80 = string[80];
var gd,gm,x,y,songay,loi: integer;
    luongngay,luongthang: real; hoten,xau: str80;

procedure InXau(var x,y: integer; s: str80);
begin OutTextxy(x,y,s); x:= x+ TextWidth(s) end;

procedure InXauXd(var x,y: integer; s: str80);
begin OutTextxy(x,y,s); x:= 10; y:= y+TextHeight(s)+5 end;

procedure DocXau(var x,y: integer; var s: str80);
```

```
var c: char;
begin  s:= '';
repeat c:= readkey;
if c <> #13 then begin InXau(x,y,c); s:=s+c end;
until c = #13;
end;

begin gd:=Vga; gm:=VgaHi; Initgraph(gd,gm,'');
bar(0,0,630,200); setcolor(blue);
SetTextStyle(0,0,1); x:=10; y:=10;
InXau(x,y,'Vao ho va ten : '); DocXau(x, y, hoten);InXauXd(x,y,' ');
InXau(x,y,'Vao so ngay cong : '); DocXau(x,y,xau);
InXauXd(x,y,' ');Val(xau,songay,loi);
InXau(x,y,'Vao luong ngay : '); DocXau(x, y, xau);
InXauXd(x,y,' '); Val(xau,luongngay,loi);
luongthang:= songay * luongngay;
InXauXd(x,y,' '); InXauXd(x,y,'Nguoi nhan luong : '+ hoten);
Str(luongthang:8:0,xau); InXauXd(x,y,'Luong thang : '+ xau + '
dong');
readkey; closegraph;
end.
```

Bài 2 chương 9 : vẽ tự do trong chế độ đồ hoạ bằng bút vẽ

```
uses crt,graph;
var gd,gm,x,y,maxx,maxy: integer; c: char;
begin gd:= detect; initgraph(gd,gm,''); setbkcolor(blue);
maxx:=getmaxx; maxy:=getmaxy; x:=maxx div 2; y:=maxy div 2;
putpixel(x, y, white);
repeat  c := readkey;
if (c = #0) then
  begin c := readkey;
    case c of
     #75: if (x>0) then begin dec(x); putpixel(x,y,white) end;
     #77: if (x<maxx-1) then begin inc(x);putpixel(x,y,white) end;
     #72: if (y>0) then begin dec(y);putpixel(x,y,white) end;
     #80: if (y<maxy-1) then begin inc(y); putpixel(x,y,white) end;
    end;
 end;
until c=#27; closegraph;
end.
```

Bài 4 chương 9 : quả bóng màu đỏ chuyển động ngẫu nhiên

```
uses graph,crt;
var gd,gm:integer; p:pointer; s:longint; i,x,x1,y1,y:integer;

begin gd:=detect; initgraph(gd,gm,'');
setfillstyle(1,red); setcolor(red); fillellipse(50,50,50,50);
s:=imagesize(0,0,100,100); getmem(p,s); getimage(0,0,100,100,p^);

delay(1000); cleardevice; setbkcolor(black); randomize; i:= 0;
repeat putpixel(random(640),random(480),random(150)); inc(i);
until i > 3000; delay(1000);

x:= 320-50; y:= 240-50; x1:= x; y1:= y;
repeat putimage(x1,y1,p^,xorput);delay(100);
putimage(x1,y1,p^,xorput);
```

```
repeat x1:= x+random(50)-25; y1:= y+random(50)-25;
until (x1>=0)and(x1<=639-100)and(y1>=0)and(y1<=479-100);
x:=x1;y:=y1;
until keypressed;  closegraph;
end.
```

Bài 7 chương 9 : vẽ đường xoáy trôn ốc

```
uses crt, graph;
var gd, gm, u, v, u1, u2, v1, v2: integer;
    x, y, x1, y1, x2, y2, t1x, t1y, t, t1, r: real;
begin clrscr; x1:=-51; x2:=51; y1:=-51; y2:=51;
u1:=30; u2:=475; v1:=450; v2:=5;
t1x:=(u2-u1)/(x2-x1); t1y:=(v1-v2)/(y2-y1);
gd:=detect; initgraph(gd,gm,''); setcolor(red);
rectangle(u1,v2,u2,v1); t:=0; t1:=pi/180;
repeat r:= 50-0.3*t*t1;
x:= r* cos(t * t1); y:= r * sin(t * t1);
u:= u1 + Round((x-x1) * t1x); v:= v1 - Round((y-y1) * t1y);
if (u>=u1) and (u<=u2) and (v>=v2) and(v<=v1) then
Putpixel(u,v,white); t:=t+0.05;
until (r<0); readln;
end.
```

Bài 10 chương 9 : chiếc đồng hồ chạy

```
uses crt,graph,dos;
var x,y,x1,y1,x2,y2,goc: integer;  goc1,goc2,goc3,old,gd,gm:integer;
    gio,phut,giay,mili:word;  so,gio1,phut1,giay1:string;  rad:real;

procedure kim_giay(goc:integer);
begin rad:=goc*pi/180; x1:=round(x+150*sin(rad));
y1:=round(y-150*cos(rad)); setlinestyle(0,0,1); line(x,y,x1,y1);
if (goc mod 6=0) then
     begin sound(3000);delay(10);nosound; end;
end;

procedure kim_phut(goc1:integer);
begin rad:=goc1*pi/180; x1:=round(x+150*sin(rad));
y1:=round(y-150*cos(rad)); setlinestyle(0,0,3); line(x,y,x1,y1);
end;

procedure kim_gio(goc2:integer);
begin rad:=goc2*pi/180;
x1:=round(x+100*sin(rad)); y1:=round(y-100*cos(rad));
setlinestyle(1,1,10); line(x,y,x1,y1);
end;

Begin gd:=detect; initgraph(gd,gm,'');

(* -------- Ve 3 vong tron ------- *)
setcolor(10); setbkcolor(white); x:=getmaxx div 2; y:=getmaxy div 2;
setfillstyle(1,10); setcolor(4);
circle(x,y,210); circle(x,y,170); circle(x,y,5); floodfill(x,y,4);
setfillstyle(2,14); floodfill(x+11,y+11,4);
setfillstyle(1,blue); floodfill(x+88,y+158,4);

(* ------- Ve so tren mat dong ho --------- *)
```

```
setcolor(12); settextstyle(0,0,2);
outtextxy(x-18,y-197,'12'); outtextxy(x+183,y-5,'3');
outtextxy(x-196,y-5,'9');   outtextxy(x-10,y+183,'6');
outtextxy(x+160,y-98,'2');  outtextxy(x-180,y-98,'10');
outtextxy(x+89,y-170,'1');  outtextxy(x-114,y-170,'11');
outtextxy(x+160,y+88,'4');  outtextxy(x-172,y+88,'8');
outtextxy(x+88,y+158,'5');  outtextxy(x-110,y+154,'7');

(* -------- Ke mac dong ho -------- *)
setcolor(12); settextstyle(0,0,1); outtextxy(x-20,y+120,'VIETNAM');

(* ----- Ve 3 kim dong ho ------- *)
setwritemode(xorput); goc:=0; goc1:=0; goc2:=0;
gettime(gio,phut,giay,mili); goc:=giay*6; old:=giay;
kim_giay(goc); goc1:=phut*6+ giay div 10; kim_phut(goc1);
goc2:=gio*30+phut div 2; kim_gio(goc2);

(* -------- Cho dong ho chay ---------- *)
repeat gettime(gio,phut,giay,mili);
if giay<>old then
  begin  kim_giay(goc); goc:=giay*6; kim_giay(goc); old:=giay;
    if phut*6+giay div 10 <>phut then
       begin kim_phut(goc1); goc1:=phut*6+giay div 10;
          kim_phut(goc1); kim_gio(goc2);
          goc2:=gio*30+phut div 2; kim_gio(goc2);
    end;
  end;
until keypressed;
closegraph;
end.
```

Bài 11 chương 9 : bóng chuyển động bật tường

```
uses graph, crt;
var h1, h2, x, y, n, gm, gd: integer; p: pointer;
begin  gd:=detect; initgraph(gd,gm,''); randomize;
setbkcolor(cyan); setcolor(red); setfillstyle(1,red);
fillellipse(20,20,20,20);
n:=imagesize(0,0,40,40); getmem(p,n); getimage(0,0,40,40,p^);
cleardevice; x:=random(550)+50; y:=random(400)+50; h1:= 1; h2 := 1;
repeat  putimage(x,y,p^,1); delay(100); putimage(x,y,p^,1);
x:= x+ 5*h1; y:= y+ 5*h2; if (x >= getmaxx-40) then h1:= -h1;
if (y >= getmaxy-40) then h2 := -h2;
if (x<=0) then h1:= -h1; if (y<=0) then h2:= -h2;
until  keypressed;
closegraph;
end.
```

Bài 12 chương 9 : tạo menu dọc trong chế độ đồ hoạ

```
uses crt,graph; label L1; const  somuc= 4;
var t: array[0..11] of string[50]; i,j,gm,gd:integer; c1,c2:char;

procedure VeThanh(i: integer);
begin setfillstyle(1,darkgray);
  bar(200,50+((i-1)*20),430,70+((i-1)*20));
  setcolor(magenta); rectangle(200,50+((i-1)*20),430,70+((i-1)*20));
```

```
  setcolor(yellow); outtextxy(215,55+((i-1)*20),t[i]);
end;

procedure VeHopSang(j: integer);
begin setcolor(white); setfillstyle(1,blue);
  bar(201,53+((j-1)*20),429,66+((j-1)*20));
  outtextxy(215,55+((j-1)*20),t[j]);
end;

procedure dohoa1;
begin cleardevice; fillellipse(320,240,150,150); readkey end;
procedure dohoa2;
begin  cleardevice; bar3d(130,200,400,290,50,true); readkey end;
procedure  dohoa3;
begin  cleardevice; fillellipse(320,240,150,80); readkey end;

Begin t[0]:='   MENU CHINH DO HOA';
t[1]:='1. Chuong trinh do hoa 1'; t[2]:='2. Chuong trinh do hoa 2';
t[3]:='3. Chuong trinh do hoa 3'; t[4]:='4. Ket thuc              ';
gd:=detect; initgraph(gd,gm,'');  setbkcolor(lightgray); j:=1;
L1: cleardevice; settextstyle(0,0,0); setfillstyle(1,cyan);
bar(200,20,430,40); setcolor(magenta); rectangle(200,20,430,40);
setcolor(yellow); outtextxy(215,25,t[0]);
for i:=1 to somuc do VeThanh(i);  VeHopSang(j);
while true do
begin  c1:= readkey; if (c1=#0) then c2:=readkey;
  if (c1=#13) and (j=somuc) then break
  else if (c1=#0) and ((c2=#80) or (c2=#72)) then
     begin if (c2=#80) then
                begin inc(j);  if (j>1) then VeThanh(j-1);
                    if (j=somuc+1) then j:=1; VeHopSang(j) end
           else begin dec(j);if (j<somuc)and(j>-1)then VeThanh(j+1);
                    if (j<1) then j:=somuc; VeHopSang(j) end
     end
  else if (c1=#13) then
          begin  case j of 1: dohoa1; 2: dohoa2; 3: dohoa3; end;
               goto L1; end;
end;
closegraph; End.
```

Bài 13 chương 9 : chùm ánh sáng chiếu vào gương

```
Uses graph,crt;
var gd,gm,r,u0,v0,u1,v1,u2,v2,x0,y0,x1,y1,x2,y2,u3,v3,x3,y3,u4,v4,
    x4,y4,u5,u6,u7,u8,v5,v6,v7,v8,x5,x6,x7,x8,y5,y6,y7,y8,xx,yy,
    dai,rong,x11,x12,x13,x14,y11,y12,y13,y14: integer;
    c: char; k: real;

procedure Doi(u,v,x0,y0: integer; var x,y: integer);
begin x:=x0+u; y:=y0-v end;

begin gd:=detect; initgraph(gd,gm,'');
setcolor(lightred); setbkcolor(cyan); K:=0.5; r:=315;
dai:=100;  rong:=60;
x11:=320-dai-30; y11:=400+30; x12:=320+dai-30; y12:=400+30;
x13:=320+dai+30; y13:=400-30; x14:=320-dai+30; y14:=400-30;
repeat
```

```
(* ---------- Ve guong ---------- *)
setfillstyle(1,white); setcolor(white); line(x11,y11,x12,y12);
line(x13,y13,x12,y12); line(x13,y13,x14,y14); line(x11,y11,x14,y14);
floodfill(320,400,white); setcolor(lightred);

(* ---------- Ve chum tia sang ---------- *)
u0:=0; v0:=0; Doi(u0,v0,320,400,x0,y0);
u1:= round(r* cos(k)); v1:=round(r* sin(k));
Doi(u1,v1,320,400,x1,y1); xx:=x1; yy:=y1; line(x0,y0,x1,y1);
u2:=-u1; v2:=v1; Doi(u2,v2,320,400,x2,y2); line(x0,y0,x2,y2);
u3:=round((r-5)*cos(k-pi/180)); v3:= round((r-5)*sin(k-pi/180));
u3:=-u3; Doi(u3,v3,320,400,x3,y3); line(x2,y2,x3,y3);
u4:=round((r-5)*cos(k+pi/180)); v4:= round((r-5)*sin(k+pi/180));
u4:=-u4; Doi(u4,v4,320,400,x4,y4); line(x2,y2,x4,y4);

Doi(u0,v0,340,400,x0,y0); Doi(u1,v1,340,400,x1,y1);
Doi(u2,v2,340,400,x2,y2); line(x0,y0,x1,y1); line(x0,y0,x2,y2);
u3:=round((r-5)*cos(k-pi/180)); v3:= round((r-5)*sin(k-pi/180));
u3:=-u3; Doi(u3,v3,340,400,x3,y3);line(x2,y2,x3,y3);
u4:=round((r-5)*cos(k+pi/180)); v4:= round((r-5)*sin(k+pi/180));
u4:=-u4; Doi(u4,v4,340,400,x4,y4); line(x2,y2,x4,y4);

Doi(u0,v0,300,400,x0,y0); Doi(u1,v1,300,400,x1,y1);
Doi(u2,v2,300,400,x2,y2); line(x0,y0,x1,y1);line(x0,y0,x2,y2);
u3:=round((r-5)*cos(k-pi/180)); v3:= round((r-5)*sin(k-pi/180));
u3:=-u3; Doi(u3,v3,300,400,x3,y3);line(x2,y2,x3,y3);
u4:=round((r-5)*cos(k+pi/180)); v4:= round((r-5)*sin(k+pi/180));
u4:=-u4;Doi(u4,v4,300,400,x4,y4);line(x2,y2,x4,y4);

{---------- Ve den chieu ----------}
u5:=round((r-20)*cos(k+5*(pi/180)));v5:=round((r-
20)*sin(k+5*(pi/180)));
u8:=round((r-20)*cos(k-5*(pi/180)));v8:=round((r-20)*sin(k-
5*(pi/180)));
Doi(u5,v5,320,400,x5,y5);Doi(u8,v8,320,400,x8,y8);

u7:=round((r+22)*cos(k+3*(pi/180)));
v7:=round((r+22)*sin(k+3*(pi/180)));
u6:=round((r+22)*cos(k-3*(pi/180)));
v6:=round((r+22)*sin(k-3*(pi/180)));
Doi(u7,v7,320,400,x7,y7);Doi(u6,v6,320,400,x6,y6);
setcolor(lightmagenta);line(x5,y5,x8,y8);line(x6,y6,x7,y7);
line(x5,y5,x7,y7);line(x6,y6,x8,y8); setfillstyle(1,lightmagenta);
floodfill(xx+2,yy+2,lightmagenta);

{---------- Viet chu ----------}
setColor(Yellow);setTextStyle(0,0,2);setTextJustify(1,2);
OutTextxy(320,30,'TRUONG THPT LY NAM DE');
OutTextxy(320,450,'GUONG');

{ Chuyen dong: an T tang goc, an G thi giam goc, an X ket thuc }
c:=readkey;
if c='t' then
  begin cleardevice;k:=k+0.05;
       if k> (pi/2)-5*(pi/180) then k:= pi/4 end;
if c='g' then
  begin cleardevice;k:=k-0.05;
```

```
       if k< 10*(pi/180) then k:= pi/4 end;
setcolor(lightred);
until c='x'; closegraph;
end.
```

Bìa 14 chương 9 : tìm bao lồi của các điểm trên mặt phẳng

Hàm Rephai(p, q, r): nếu r nằm bên phải nửa đường thẳng đi từ p qua q thì hàm cho giá trị True, trái lại cho False. Đầu tiên p chứa n điểm nhập vào từ tệp văn bản BaoLoi.inp (dòng đầu chứa n, các dòng dưới chứa toạ độ (x,y) của từng điểm), tiếp theo chương trình sắp xếp mảng p tăng dần theo toạ độ x (nếu x trùng nhau thì sắp xếp theo giá trị y tăng dần): p[1], p[2], . . . , p[n]. Sau đó chương trình tìm các đỉnh ở phía trên của bao lồi theo chiều kim đồng hồ đi từ p[1] đến p[n] lưu vào mảng t, tìm tiếp các đỉnh ở phía dưới của bao lồi theo chiều kim đồng hồ đi từ p[n] đến p[1] lưu vào mảng d, cuối cùng ghi các đỉnh của bao lồi theo chiều kim đồng hồ vào tệp văn bản BaoLoi.out (dòng đầu ghi số đỉnh của bao lồi, các dòng sau ghi toạ độ các đỉnh). Từ tệp Baoloi.int và Baoloi.out ta có thể vẽ bao lồi và các điểm ban đầu trên màn hình đồ hoạ.

```
type  diem = record  x,y:real end;
      mangdiem = array[1..500] of diem;
var  p,t,d: mangdiem;  tg: diem;  i,j,k,h,n: integer;
     f1,f2: text;

function Rephai(p,q,r:diem): boolean;
begin if (q.x*r.y+p.x*q.y+p.y*r.x-q.x*p.y-p.x*r.y-r.x*q.y < 0) then
          Rephai:= True else Rephai := False;
end;

begin
assign(f1,'BaoLoi.inp'); reset(f1);
assign(f2,'BaoLoi.out');  rewrite(f2);
readln(f1,n);
for i:=1 to n do  read(f1,p[i].x,p[i].y);  close(f1);
for i:=1 to n-1 do   for j:=n-1 downto i do
 if (p[j].x>p[j+1].x)or((p[j].x=p[j+1].x)and(p[j].y>p[j+1].y)) then
            begin tg:=p[j];  p[j]:=p[j+1];  p[j+1]:=tg   end;
t[1]:= p[1]; t[2]:= p[2]; k:=2;
for i:=3 to n do
    begin  inc(k);  t[k]:= p[i];
       while (k>2) and (not Rephai(t[k-2], t[k-1], t[k])) do
                 begin  t[k-1]:= t[k]; dec(k) end;
    end;

d[1]:= p[n]; d[2]:= p[n-1]; h:=2;
for i:= n-2 downto 1 do
   begin  inc(h);  d[h]:=p[i];
       while (h>2)  and (not Rephai(d[h-2],d[h-1],d[h])) do
         begin  d[h-1]:=d[h]; dec(h)  end
   end;

writeln(f2,k+h-2);
for i:=1 to k do writeln(f2,t[i].x:13:5,'    ',t[i].y:13:5);
for i:=2 to h-1 do writeln(f2,d[i].x:13:5,'    ',d[i].y:13:5);
close(f2);
```

```
write('Xong.....'); readln;
end.
```

Ví dụ tệp Baoloi.inp gồm 21 điểm:

```
21
2 2    2 1   4 2   0 2   1 2   2 3   3 0   3 1   3 2   1   0.5
1 1    4 1   1 3   3 3   4 3   1 4   2 4   3 4   4 4   5   4     3 5
```

Tệp Baoloi.out chứa các đỉnh của bao lồi liệt kê theo chiều kim đồng hồ:

```
7
0.00000          2.00000
1.00000          4.00000
3.00000          5.00000
5.00000          4.00000
4.00000          1.00000
3.00000          0.00000
1.00000          0.50000
```

Bài 9 chương 12 : xếp n quân hậu trên bàn cờ

Đánh số cột và dòng của bàn cờ từ 1 đến n, mỗi dòng được xếp đúng 1 quân hậu. Vấn đề còn lại là xem mỗi quân hậu được xếp đúng vào cột nào, do đó một cách xếp được biểu diễn bằng bộ (x[1], x[2], . . ., x[n]), trong đó x[k] = j có nghĩa là quân hậu dòng k xếp vào cột j. Thành phần x[k] có thể nhận giá trị từ 1 đến n, giá trị j là được chấp nhận nếu ô (k,j) chưa bị các quân hậu trước chiếu đến (quân hậu có thể ăn ngang, dọc và hai đường chéo). Ta cần ghi nhận trạng thái của bàn cờ trước cũng như sau khi xếp một quân hậu.

Việc kiểm soát theo chiều ngang là không cần thiết vì mỗi dòng chỉ xếp một quân hậu. Việc kiểm soát theo chiều dọc được ghi nhận nhờ dãy biến logic a[j] với quy ước a[j] = true nếu cột j còn trống. Đối với hai đường chéo ta nhận xét rằng một đường có phương trình là k + j = const, còn đường kia là k - j = const (2 <= k + j <= 2n, 1 - n <= k - j <= n -1), từ đó đường chéo thứ nhất được ghi nhận nhờ biến logic b[j] (2 <= j <= 2n), và đường chéo thứ hai nhờ biến c[j] (1- n <= j <= n -1) với quy ước các đường này còn trống nếu biến tương ứng có giá trị True. Lúc đầu các biến trạng thái a[j], b[j], c[j] đều nhận giá trị True. Như vậy giá trị j được chấp nhận nếu cả ba biến a[j], b[k+j], c[k-j] cùng có giá trị True. Các biến này cần gán lại False khi xếp xong quân hậu thứ k và trả lại True sau khi gọi InKetQua hay QuanHau(k+1).

```
var n: integer; d: word;
    x: array[1..20] of integer;
    a: array[1..20] of boolean;
    b: array[2..40] of boolean;
    c: array[-19..19] of boolean;

procedure KhoiGan;
var i: integer;
begin write('n = '); readln(n);
d:= 0; for i:=1 to n do a[i]:= true;
for i:=2 to 2*n do b[i]:= true;
for i:= 1-n to n-1 do c[i]:= true;
end;

procedure InKetQua;
```

```
var i: integer;
begin d:= d+1; write(d:5,': ');
for i:=1 to n do write(x[i]:3); writeln;
if d mod 23=0 then readln;
end;

procedure QuanHau(k: integer);
var j: integer;
begin
for j:=1 to n do
if a[j] and b[k+j] and c[k-j] then
  begin x[k]:=j; a[j]:= false;
        b[k+j]:= false; c[k-j]:= false;
        if k=n then InKetQua else QuanHau(k+1);
        a[j]:= true; b[k+j]:= true; c[k-j]:= true;
  end;
end;
BEGIN KhoiGan;
QuanHau(1);
write('Xong...'); readln;
END.
```

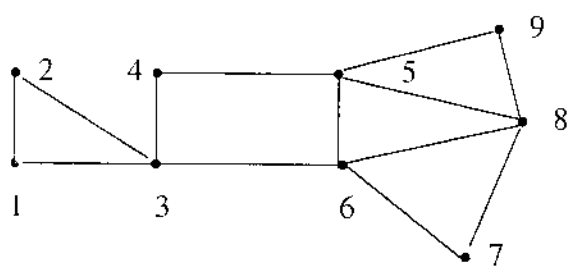
Từ chương trình nhận được các cách sắp xếp 5 quân hậu trên bàn cờ 5 x 5 như sau: (13524) (14253) (24135) (25314) (31425) (35241) (41352) (42531) (52413) (53142).

Bài 2 chương 13 : chu trình Euler

```
uses crt;
label l1;
const nmax=30;
type mang1= array[1..nmax,1..nmax] of byte;
     mang2= array[1..nmax] of boolean;
     mang3= array[1..nmax] of integer;
var c: mang1;  lv: mang2;
    i,j,u,n,bu,dem: integer; f: text;

function lienth(u,v: integer; lv: mang2): integer;
var i,j,d,k,l: integer;   p: mang3;
begin
c[u,v]:=0; c[v,u]:=0;
for i:=1 to n do p[i]:=0; d:=0;
for i:=1 to n do
 begin   if (p[i]=0) and lv[i] then
     begin  inc(d);   p[i]:=d;
       for k:=1 to n do  for j:=1 to n do  for l:=1 to n do
       if (p[j]=0)and lv[j] and(p[l]=d)and(c[l,j]=1) then p[j]:=d;
     end;
 end;
c[u,v]:=1; c[v,u]:=1; lienth:=d;
end;

Begin clrscr;
assign(f,'Euler.dat'); reset(f);
readln(f,n);
for i:=1 to n do for j:=1 to n do read(f,c[i,j]);
close(f);
```

```
write('Bat dau di tu dinh u = '); readln(u);
writeln('Duong di Euler tim duoc : '); dem:=0;
for j:=1 to n do lv[j]:= true;
l1: bu:=0;
for j:=1 to n do if c[u,j]=1 then inc(bu); dem:=dem+1;
if bu=1 then
 begin for j:=1 to n do if c[u,j]=1 then
         begin  lv[u]:= false; c[u,j]:=0; c[j,u]:=0;
          writeln('chon canh ',dem:3,' la: ',u:2,' - ',j:2);
          u:= j; goto l1;
         end;
 end else
 begin
  for j:=1 to n do if c[u,j]=1 then
    begin if lienth(u,j,lv)=1 then
      begin  c[u,j]:=0; c[j,u]:=0;
        writeln('chon canh ',dem:3,' la: ',u:2,' - ',j:2);
        u:= j;  goto l1;
      end
    end
 end;
readln; END.
```

Dữ liệu về đồ thị để trong tệp Euler.dat gồm có: n - số đỉnh của đồ thị, ma trận kề C[n,n] được ghi theo hàng. Khi chạy chương trình ta phải nhập vào đỉnh xuất phát của đường đi Euler, máy sẽ in ra lần lượt các cạnh của đường đi Euler. Trong chương trình sử dụng một hàm là LienTh(u,v,lv). Trong quá trình tìm đường đi Euler, đồ thị ban đầu bị xoá các đỉnh và các cạnh dần dần. Hàm LienTh(u,v,lv) cho biết số thành phần liên thông của đồ thị tại thời điểm đang xét, nếu ta xoá cạnh (u,v) của đồ thị.



Với đồ thị cho ở hình trên thì tệp số liệu Euler.dat có dạng:

```
9
0 1 1 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0
1 1 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0 0
0 0 0 1 0 1 0 1 1
0 0 1 0 1 0 1 1 0
0 0 0 0 0 1 0 1 0
0 0 0 0 1 1 1 0 1
0 0 0 0 1 0 0 1 0
```

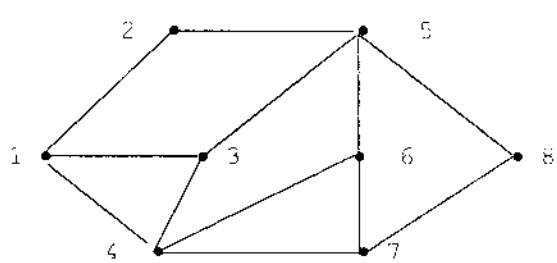
Kết quả nhận được đường đi Euler nếu xuất phát từ đỉnh 1 là: (1 - 2) (2 - 3) (3 - 4) (4 - 5) (5 - 6) (6 - 7) (7 - 8) (8 - 5) (5 - 9) (9- 8) (8 - 6) (6 - 3) (3 - 1).

Bài 3 chương 13 : duyệt tất cả các chu trình Hamilton

```
var i, j, n: integer;
    c: array[1..10,1..10] of byte;
    p: array[1..10] of byte;
    b: array[1..10] of boolean;
    d: word; f: text;

procedure xuly;
label 11;
var t: integer; kd: boolean;
begin   kd:=True;
   for t:=1 to n-1 do
            if c[p[t],p[t+1]] = 0 then
                          begin kd:= false; goto 11; end;
   if c[p[n],p[1]]=0 then kd:= False;
11:if kd then
    begin  d:=d+1;
      write('Chu trinh Hamilton ',d:4,' la : ');
      for t:=1 to n do write(p[t]:3); writeln;
      if d mod 23 = 0 then readln;
    end;
end;

procedure Hamilton(k: integer);
var j,i1:integer;
begin for j:=1 to n do  if b[j] then
     begin p[k]:=j; b[j]:= False;
           if k=n then  xuly else Hamilton(k+1); b[j]:= True;
     end;
end;

Begin
assign(f,'hamil.dat'); reset(f); readln(f,n);
for i:= 1 to n do for j:=1 to n do read(f,c[i,j]); close(f);
for i:=1 to n do b[i]:=True; d:=0; Hamilton(1); readln;
End.
```

Mảng P[1..n] lưu các hoán vị sinh ra của tập { 1, 2, . . ., n }. Thủ tục Hamilton(k) dùng đệ quy thể hiện thuật toán quay lui. Thủ tục XuLy dùng để kiểm tra một hoán vị có phải là đường đi Hamilton hay không. Số liệu vào để trong tệp Hamil.dat gồm các thông tin: n - số đỉnh của đồ thị, ma trận kề C[1..n,1..n] viết theo từng hàng. Với đồ thị như ở Hình dưới thì tệp số liệu có dạng:

```
8
0 1 1 1 0 0 0 0
1 0 0 0 1 0 0 0
1 0 0 1 1 0 0 0
1 0 1 0 0 1 1 0
0 1 1 0 0 1 0 1
0 0 0 1 1 0 1 0
0 0 0 1 0 1 0 1
0 0 0 0 1 0 1 0
```



Khi chạy máy liệt kê 16 chu trình Hamilton tìm được là:

```
Chu trinh Hamilton 01 la : 1 2 5 8 7 6 4 3
Chu trinh Hamilton 02 la : 1 3 4 6 7 8 5 2  . . . . . .
Chu trinh Hamilton 16 la : 8 7 6 4 3 1 2 5
```

Bài 4 chương 13 : đường đi ngắn nhất giữa hai đỉnh

```
label 11;
const sdmax=30;  scmax=100;
type mang1=array[1..sdmax] of integer;
     mang2=array[1..scmax] of integer;
     mang3=array[1..sdmax] of real;
     mang4=array[1..scmax] of real;
var dtr,ddi:mang1; c1,c2,xc:mang2; nh:mang3; ddai,cn:mang4;
    i,j,it,sd,sc,sc1,goc,dich:integer; mindd:real; f1,f2:text;

procedure Ddnn(goc,dich,sc1,sc,sd:integer; c1,c2:mang2;
  ddai:mang4;nh:mang3;dtr:mang1; xc:mang2;cn:mang4;
  var ddi:mang1;var mindd:real;var it:integer);
var kn,i,k,k1,k2,inhnh:integer; nhnh:real;
begin
for i:=1 to sd do  begin nh[i]:=-1.0; dtr[i]:=-1; end;
nh[goc]:=0.0; dtr[goc]:=0; it:=0;
while (true) do

begin        (* bat dau vong lap theo while *)
it:=it+1; writeln('Buoc lap ',it);
for i:=1 to sc do begin xc[i]:=0; cn[i]:=1.0e+30; end;

(* Tim moi cung noi ra ngoai tep da gan nhan *)
for k:=1 to sd do if (nh[k]>-1) then
begin

(* Mo dau chu trinh theo k: tinh xc[.], cn[.] doi voi nhung canh
hoac cung di tu dinh da gan nhan k toi cac dinh chua gan nhan *)
for i:=1 to sc do if (c1[i]=k) and (nh[c2[i]]<-0.5) then
         begin  xc[i]:=1; cn[i]:=nh[c1[i]]+ddai[i]; end;
for i:=1 to sc1 do
    if (c2[i]=k) and (nh[c1[i]]<-0.5) then
        begin  xc[i]:=-1; cn[i]:=nh[c2[i]]+ddai[i]; end;
end;  (* Ket thuc vong lap theo k *)
nhnh:=cn[1]; inhnh:=1;
for i:=2 to sc do if (cn[i]< nhnh) then
            begin nhnh:=cn[i]; inhnh:=i end;

(* Gan nhan va danh dau dinh dan toi dinh duoc gan nhan *)
if (xc[inhnh]=1) then
   begin kn:=c2[inhnh]; nh[kn]:=cn[inhnh];
         dtr[kn]:=c1[inhnh]; end
else begin kn:=c1[inhnh]; nh[kn]:=cn[inhnh];
          dtr[kn]:=c2[inhnh]; end;
if (kn=dich) then
 begin   mindd:=nh[dich];  for i:=1 to sd do ddi[i]:=0;
   i:=1; k1:=dich; k2:=dtr[dich];
   repeat  begin  ddi[sd-i+1]:=k1; i:=i+1; k1:=k2;
                  if (k1<>0) then k2:= dtr[k1]; end;
   until (k1=0);
   exit;
```

```
 end;
end;   (* het vong lap theo while *)
end;

Begin
assign(f1,'ddnn.inp'); reset(f1);
assign(f2,'ddnn.out'); rewrite(f2);
read(f1,sc1,sc,sd);
for i:=1 to sc do read(f1,c1[i],c2[i],ddai[i]);
writeln(f2,'So cung vo huong: ',sc1);
writeln(f2,'Tong so cung: ',sc); writeln(f2,'So dinh: ',sd);
for i:=1 to sc do
writeln(f2,'Cung ',i,': tu ',c1[i],' den ',c2[i],
          ' dai ',ddai[i]:1:4);
close(f1);

l1: write('Noi xuat phat la dinh: '); readln(goc);
write('Noi den la dinh      : '); readln(dich);
writeln(f2);
writeln(f2,'Dinh xuat phat : ',goc);
writeln(f2,'Noi den la dinh: ',dich);
Ddnn(goc,dich,sc1,sc,sd,c1,c2,ddai,nh,dtr,xc,cn,ddi,mindd,it);

writeln(f2,'Do dai duong di ng.nhat: ',mindd:1:5);
writeln(f2,'Duong di ng.nhat qua cac dinh:');
writeln('Do dai duong di ng.nhat: ',mindd:1:5);
writeln('Duong di ng.nhat di qua cac dinh: '); j:=1;
for i:=1 to sd do  if (ddi[i]>0) then
    begin   write(f2,'  ',ddi[i]); write('  ',ddi[i]);
            if (j mod 8=0) then begin  writeln(f2); writeln ;  end;
            j:=j+1;
    end;
writeln(f2); writeln; write('Tiep tuc tim DDNN 1/0: '); readln(i);
if (i=1) then goto l1; close(f2);
End.
```

Thủ tục **Ddnn** có các thông số sau:

- Goc: tên đỉnh nguồn. Dich: tên đỉnh đích. SC1: tổng số cạnh vô hướng. SC: tổng số cạnh vô hướng và cung có hướng. SD: số lượng đỉnh.

- Các mảng C1[sc], C2[sc], DDAI[sc] có ý nghĩa: cạnh hay cung nối đỉnh C1[i] với đỉnh C2[i] có độ dài là DDAI[i].

- Mảng NH[sd]: nhãn của đỉnh i là NH[i]. Mảng DTR[sd] (viết tắt của "đỉnh trước"): đỉnh i được gán nhãn từ đỉnh DTR[i]. Mảng DDI[sd]: lưu trữ đường đi ngắn nhất tìm được.

- Mảng XC[sc] và CN[sc]: các mảng trung gian. MINDD: độ dài đường đi ngắn nhất. IT: số bước lặp (số đỉnh được gán nhãn).

Dữ liệu về đồ thị cho trong tệp văn bản **Ddnn.inp** gồm các thông tin: sc1, sc, sd và c1[i], c2[i], ddai[i] với i = 1, . . . , sc. Với đồ thị như trong Hình 4 Mục 4 thì tệp số liệu có dạng sau:

```
12 15 8
1 2 3 1 3 5 1 4 2
2 3 1 2 5 7 3 4 4
```

```
3 5 5 4 6 3 5 8 3
6 7 4 6 8 6 7 8 5
6 3 1 6 5 2 7 4 6
```

Kết quả tìm đường đi ngắn nhất từ đỉnh 1 tới đỉnh 8 và đường đi ngắn nhất từ đỉnh 8 về đỉnh 1 cho trong tệp **Ddnn.out** như sau:

```
So cung vo huong: 12
Tong so cung: 15
So dinh: 8
Cung 1: tu 1 den 2 dai 3.0000
Cung 2: tu 1 den 3 dai 5.0000
Cung 3: tu 1 den 4 dai 2.0000
Cung 4: tu 2 den 3 dai 1.0000
C+ung 5: tu 2 den 5 dai 7.0000
Cung 6: tu 3 den 4 dai 4.0000
Cung 7: tu 3 den 5 dai 5.0000
Cung 8: tu 4 den 6 dai 3.0000
Cung 9: tu 5 den 8 dai 3.0000
Cung 10: tu 6 den 7 dai 4.0000
Cung 11: tu 6 den 8 dai 6.0000
Cung 12: tu 7 den 8 dai 5.0000
Cung 13: tu 6 den 3 dai 1.0000
Cung 14: tu 6 den 5 dai 2.0000
Cung 15: tu 7 den 4 dai 6.0000
Dinh xuat phat : 1
Noi den la dinh: 8
Do dai duong di ng.nhat: 10.00000
Duong di ngan nhat qua cac dinh:
  1 4 6 5 8
Dinh xuat phat : 8
Noi den la dinh: 1
Do dai duong di ng.nhat: 11.00000
Duong di ngan nhat qua cac dinh:
   8 6 3 2 1
```

Mục lục

BẠN ĐỌC LƯU Ý. Để tránh chọn nhầm sách giả, các bạn lưu ý đặc điểm nhận dạng sách thật của cuốn "Giáo trình Turbo Pascal 7.0" : chữ màu xanh in sắc nét, cuối "Lời mở đầu" trang 2 có chữ ký bằng tay của tác giả bằng bút bi đỏ theo mẫu ở dưới. Đặc điểm sách giả: chữ không sắc nét và mờ; trang 2 không có chữ ký của tác giả hoặc chữ ký giả khác xa so với chữ ký thật hoặc chữ ký được in hàng loạt bằng máy in; sách giả có nội dung sai và thiếu rất nhiều, các chương trình máy tính không chạy do được đánh máy lại.

GIÁO TRÌNH TURBO PASCAL 7.0

- Chịu trách nhiệm xuất bản: Tiến sỹ Nguyễn Xuân Thuỷ
- Cuốn sách do Bùi Thị Nhung giữ bản quyền
- In 1500 cuốn khổ 19 x 27 cm tại Công ty in Thống nhất
- Giấy chấp nhận đề tài số 105 /XB-QLXB của Cục xuất bản ngày 09-
- In xong và nộp lưu chiểu Qúy 4 năm 2004

8017295

GT Turbo Pascal 7.0
BTT02T3
Giá: 34000 VND

Giá : 34.000 đồng