SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

GIÁO TRÌNH

Thuật toán và kỹ thuật lập trình

PASCAL

DÙNG TRONG CÁC TRƯỜNG TRUNG HỌC CHUYÊN NGHIỆP

NHÀ XUẤT BẢN HÀ NỘI

http://tieulun.hopto.org

SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

NGUYỄN CHÍ TRUNG *(Chủ biên)*

GIÁO TRÌNH
THUẬT TOÁN
VÀ KỸ THUẬT LẬP TRÌNH PASCAL

(Dùng trong các trường THCN)

NHÀ XUẤT BẢN HÀ NỘI - 2005

http://tieulun.hopto.org

Mã số XB: $\frac{373 - 373.7}{HN - 05}$ 198/290/05

http://tieulun.hopto.org

Lời giới thiệu

Nước ta đang bước vào thời kỳ công nghiệp hóa, hiện đại hóa nhằm đưa Việt Nam trở thành nước công nghiệp văn minh, hiện đại.

Trong sự nghiệp cách mạng to lớn đó, công tác đào tạo nhân lực luôn giữ vai trò quan trọng. Báo cáo Chính trị của Ban Chấp hành Trung ương Đảng Cộng sản Việt Nam tại Đại hội Đảng toàn quốc lần thứ IX đã chỉ rõ: "Phát triển giáo dục và đào tạo là một trong những động lực quan trọng thúc đẩy sự nghiệp *công nghiệp hóa, hiện đại hóa*, là điều kiện để phát triển nguồn lực con người - yếu tố cơ bản để phát triển xã hội, tăng trưởng kinh tế nhanh và bền vững".

Quán triệt chủ trương, Nghị quyết của Đảng và Nhà nước và nhận thức đúng đắn về tầm quan trọng của chương trình, giáo trình đối với việc nâng cao chất lượng đào tạo, theo đề nghị của Sở Giáo dục và Đào tạo Hà Nội, ngày 23/9/2003, Ủy ban nhân dân thành phố Hà Nội đã ra Quyết định số 5620/QĐ-UB cho phép Sở Giáo dục và Đào tạo thực hiện đề án biên soạn chương trình, giáo trình trong các trường Trung học chuyên nghiệp (THCN) Hà Nội. Quyết định này thể hiện sự quan tâm sâu sắc của Thành ủy, UBND thành phố trong việc nâng cao chất lượng đào tạo và phát triển nguồn nhân lực Thủ đô.

Trên cơ sở chương trình khung của Bộ Giáo dục và Đào tạo ban hành và những kinh nghiệm rút ra từ thực tế đào tạo, Sở Giáo dục và Đào tạo đã chỉ đạo các trường THCN tổ chức biên soạn chương trình, giáo trình một cách khoa học, hệ

http://tieulun.hopto.org

thống và cập nhật những kiến thức thực tiễn phù hợp với đối tượng học sinh THCN Hà Nội.

Bộ giáo trình này là tài liệu giảng dạy và học tập trong các trường THCN ở Hà Nội, đồng thời là tài liệu tham khảo hữu ích cho các trường có đào tạo các ngành kỹ thuật - nghiệp vụ và đông đảo bạn đọc quan tâm đến vấn đề hướng nghiệp, dạy nghề.

Việc tổ chức biên soạn bộ chương trình, giáo trình này là một trong nhiều hoạt động thiết thực của ngành giáo dục và đào tạo Thủ đô để kỷ niệm "50 năm giải phóng Thủ đô", "50 năm thành lập ngành" và hướng tới kỷ niệm "1000 năm Thăng Long - Hà Nội".

Sở Giáo dục và Đào tạo Hà Nội chân thành cảm ơn Thành ủy, UBND, các sở, ban, ngành của Thành phố, Vụ Giáo dục chuyên nghiệp Bộ Giáo dục và Đào tạo, các nhà khoa học, các chuyên gia đầu ngành, các giảng viên, các nhà quản lý, các nhà doanh nghiệp đã tạo điều kiện giúp đỡ, đóng góp ý kiến, tham gia Hội đồng phản biện, Hội đồng thẩm định và Hội đồng nghiệm thu các chương trình, giáo trình.

Đây là lần đầu tiên Sở Giáo dục và Đào tạo Hà Nội tổ chức biên soạn chương trình, giáo trình. Dù đã hết sức cố gắng nhưng chắc chắn không tránh khỏi thiếu sót, bất cập. Chúng tôi mong nhận được những ý kiến đóng góp của bạn đọc để từng bước hoàn thiện bộ giáo trình trong các lần tái bản sau.

GIÁM ĐỐC SỞ GIÁO DỤC VÀ ĐÀO TẠO

http://tieulun.hopto.org

Lời nói đầu

Việc rèn luyện tư duy thuật toán góp phần bồi dưỡng cho chúng ta phẩm chất của con người lao động mới trong xã hội văn minh thông tin. Trong những phẩm chất đó phải kể đến thói quen tự kiểm tra, thói quen tìm kiếm cách giải quyết vấn đề trên máy tính một cách khoa học, tiết kiệm thời gian mà lại có hiệu quả.

Giáo trình "Thuật toán và kỹ thuật lập trình Pascal" lấy mục đích rèn luyện tư duy thuật toán làm trọng tâm, do đó chỉ cung cấp kiến thức ở mức cần thiết (về các cấu trúc dữ liệu cơ bản và các cấu trúc điều khiển) của ngôn ngữ Pascal để đủ minh họa cho việc thể hiện các thuật toán và chú trọng rèn luyện kỹ năng cài đặt thuật toán trên ngôn ngữ Pascal. Giáo trình sẽ không đề cập đến những vấn đề chuyên sâu như: Con trỏ và cấp phát động; các cấu trúc dữ liệu trừu tượng; lập trình đồ họa, âm thanh; thư viện chương trình riêng;... Để đảm bảo tính đầy đủ ở mức cần thiết, một số nội dung có thể được trình bày thêm thành bài, mục đọc thêm để tiện tham khảo.

Về cấu trúc hình thức, giáo trình chia thành các chương. Mỗi chương bao gồm một số phần kiến thức được gán chỉ số la mã. Cuối mỗi phần kiến thức đều có một số câu hỏi và bài tập để luyện tập, củng cố kiến thức.

Hệ thống các câu hỏi và bài tập của các khối kiến thức trong một chương được đánh chỉ mục liên tục. Chẳng hạn "Bài 1.15" nghĩa là bài tập thứ 15 của chương 1. Tương tự như thế, tất cả các ví dụ cùng loại (về thuật toán và chương trình) trong cùng một chương cũng được gán chỉ mục liên tục. Chẳng hạn "Ví dụ 2.14" nghĩa là ví dụ thứ 14 của chương 2. Các ví dụ thuộc loại khác (như ví dụ để minh hoạ cho một khái niệm, một quy tắc cú pháp...) thì không được gán chỉ mục như trên. Các ví dụ thuộc loại này, trong mục kiến thức đang trình bày, sẽ không được gán số thứ tự hoặc có số thứ tự, tuỳ theo mục kiến thức này có một hay có nhiều ví dụ.

Ngôn ngữ Pascal mà chúng ta sử dụng đi cùng với giáo trình này là Borland Pascal 7.x.

http://tieulun.hopto.org

Tác giả xin bày tỏ lòng biết ơn chân thành tới: PGS - TS. Nguyễn Xuân Huy (Viện Công nghệ thông tin), PSG - TS. Đoàn Văn Ban (Viện Công nghệ thông tin), GS. Phạm Văn Ất (Đại học Giao thông), TS. Dương Tử Cường (Học viện kỹ thuật quân sự), TS. Phó Đức Toàn (Đại học Sư phạm Hà Nội 1) đã có những ý kiến đóng góp quý báu để cuốn giáo trình được hoàn chỉnh và có chất lượng hơn.

Tác giả cũng xin chân thành cảm ơn sự tổ chức, hợp tác và giúp đỡ nhiệt tình của ban giám hiệu, phòng đào tạo và tổ chuyên môn trường THKT Tin học Hà Nội - ESTIH.

Tác giả cũng xin chân thành cảm ơn các bạn bè đồng nghiệp đã quan tâm và có những ý kiến đóng góp trong quá trình tác giả biên soạn giáo trình.

Tác giả

http://tieulun.hopto.org

Chương 1

CÁC VẤN ĐỀ CƠ BẢN CỦA THUẬT TOÁN

I. THUẬT TOÁN VÀ MỘT SỐ TÍNH CHẤT CƠ BẢN

1. Khái niệm thuật toán

Thuật toán (algorithm) là một dãy các chỉ thị được viết theo một quy tắc nhất định để theo đó máy tính thực hiện, sao cho, từ dữ liệu vào là giả thiết của bài toán, sau một số hữu hạn bước sẽ thu được dữ liệu ra là kết quả cần tìm của bài toán đó.

Ví dụ 1.1: Xây dựng thuật toán tìm giá trị lớn nhất trong 3 số a,b,c cho trước, kết quả gán cho biến m.

Bước 1: Nhập a,b,c;
Bước 2: Gán m := a;
Bước 3: Nếu m<b thì m := b;
Bước 4: Nếu m<c thì m := c;
Bước 5: In kết quả (số lớn nhất có giá trị là m).
Bước 6: Kết thúc.

Nhận xét:

Cuối một thuật toán luôn có bước ghi nhận kết thúc thuật toán.

Ta sử dụng ký pháp := để biểu thị một câu lệnh gán. Câu lệnh gán thực hiện việc gán giá trị của một biểu thức xác định cho một biến. Cần phân biệt câu lệnh gán và phép so sánh. Ví dụ, viết m := 5 nghĩa là biểu thị lệnh gán cho biến m bằng 5, viết m = 5 nghĩa là biểu thị việc so sánh giá trị của m với 5.

Dữ liệu kiểm thử là một bộ giá trị cụ thể của dữ liệu vào. Một tập hợp các dữ liệu kiểm thử gọi là đầy đủ nếu nó phủ kín tất cả các trường hợp của dữ liệu vào.

Ví dụ 1.1 có bộ dữ liệu kiểm thử đầy đủ gồm 6 dữ liệu kiểm thử.

http://tieulun.hopto.org

Rõ ràng, để chứng minh một thuật toán là đúng đắn thì ngoài phương pháp chứng minh bằng toán học, chúng ta có thể chứng minh bằng cách chỉ ra rằng thuật toán luôn đúng đối với bất kỳ dữ liệu kiểm thử nào trong một bộ dữ liệu kiểm thử đầy đủ.

Để chỉ ra một thuật toán sai, ta chỉ cần chỉ ra một dữ liệu kiểm thử để chỉ ra rằng thuật toán cho kết quả sai.

Bảng mô phỏng: Là một bảng dùng để diễn tả quá trình thực hiện một thuật toán trên một dữ liệu kiểm thử cụ thể. Người ta thường dùng bảng mô phỏng để giải thích thuật toán, lấy ví dụ cho thuật toán. Bảng mô phỏng thường gồm một số cột, mỗi một cột dành riêng cho một biến và các hàng để biểu diễn các bước thực hiện thuật toán, theo dõi sự thay đổi giá trị của biến đó khi thuật toán được thực hiện.

Bảng mô phỏng đầy đủ các biến cho ví dụ 1.1 với dữ liệu kiểm thử (a,b,c) tương ứng là (3,4,7) như sau:

Bước	a	b	c	m
1	3	4	7	
2	3	4	7	3
3	3	4	7	4
4	3	4	7	7
5	3	4	7	In kq 7

Để thuận lợi, bảng mô phỏng trên có thể bớt đi các cột dành cho các biến a,b,c và nói chung các biến có giá trị không bị thay đổi trong suốt quá trình thực hiện theo thuật toán thì có thể không cần đưa vào bảng mô phỏng. Thay vào việc bớt đi các cột không cần thiết, ta nên đưa vào các cột biểu thị giá trị của các điều kiện (biểu thức logic) để dễ thấy được các quyết định (các lệnh) khác nhau khi giá trị điều kiện thay đổi theo từng bước của thuật toán.

2. Một số tính chất cơ bản của thuật toán

2.1. Tính dừng

Một thuật toán phải có tính dừng, nghĩa là nó phải đảm bảo dừng lại sau một số hữu hạn bước.

http://tieulun.hopto.org

Ví dụ 1.2: Thuật toán điều khiển (mức tổng quát) hoạt động "đi" của một ro-bot như sau là vi phạm tính dừng:

Bước 1: Đứng lên;
Bước 2: Bước chân phải;
Bước 3: Bước chân trái;
Bước 4: Quay về Bước 2;
Bước 5: Kết thúc.

Nhận xét: Thuật toán ở ví dụ 1.2 trên đây không dừng vì bước 5 không bao giờ thực hiện được.

2.2. Tính đúng đắn

Một thuật toán phải có tính đúng đắn, nghĩa là nó luôn cho cùng kết quả đúng đối với một bộ dữ liệu kiểm thử đầy đủ.

Ví dụ 1.3: Nhận xét xem thuật toán sau đây thực hiện lại yêu cầu của ví dụ 1.1 có đảm bảo tính xác định không.

Bước 1: Nhập a,b,c;
Bước 2: Gán m:= a;
Bước 3: Nếu a<b thì m:= b;
Không thì Nếu a<c thì m:= c;
Bước 4: In kết quả (số lớn nhất có giá trị là m);
Bước 5: Kết thúc.

Nhận xét: Thuật toán trên sai khi a<b<c, vì vậy nó vi phạm tính đúng đắn.

2.3. Tính phổ dụng

Một thuật toán gọi là có tính phổ dụng nếu nó giải quyết được một lớp bài toán. Mức độ phổ dụng của thuật toán phụ thuộc vào tính tổng quát của thuật toán đó. Tuy nhiên, trong lập trình, đôi khi người ta giảm mức tổng quát của bài toán để làm tăng tính khả thi của chương trình (đảm bảo thời gian thực hiện, quá trình thiết kế...)

Ví dụ 1.4: Xây dựng thuật toán giải bất phương trình:

$ax + b \leq 0$ với $\forall$ a,b.

http://tieulun.hopto.org

Bước 1: Nhập a, b;
Bước 2: Nếu a=0 Thì
Nếu b≤0 thì Inkq (BPT VĐ)
Không Thì Inkq (BPT VN)
Không thì
Nếu a>0 thì Inkq (x ≤ -b/a)
Không thì Inkq (x≥ -b/a);
Bước 3: Kết thúc.

3. Câu hỏi và bài tập

Bài 1.1

Khái niệm thuật toán? Thuật toán có những tính chất cơ bản nào? Cho ví dụ minh hoạ?

Bài 1.2

Khái niệm dữ liệu kiểm thử? Thế nào là bộ dữ liệu kiểm thử đầy đủ? Hãy cho một bộ dữ liệu kiểm thử đầy đủ khi cần xem xét một thuật toán giải bất phương trình: ax + b >= 0.

Bài 1.3

Lập bảng mô phỏng để thực hiện thuật toán sau đây, sau đó hãy cho biết thuật toán giải quyết vấn đề gì?

Bước 1: Nhập a,b;
Bước 2: Gán x:= a;
Bước 3: Gán a:= b;
Bước 4: Gán b:= x;
Bước 5: In kết quả (a, b);
Bước 6: Kết thúc.

II. CÁC CẤU TRÚC ĐIỀU KHIỂN TRONG THUẬT TOÁN

1. Các hình thức mô tả thuật toán

1.1. Sử dụng hình thức mô tả theo từng bước

Là phương pháp liệt kê từng bước giống như các ví dụ 1.1, 1.2, 1.3 và 1.4 đã được trình bày.

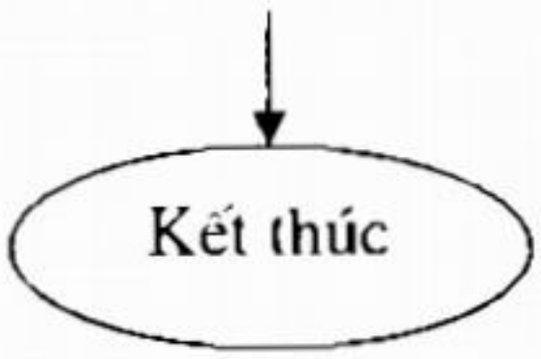
http://tieulun.hopto.org

1.2. Sử dụng sơ đồ khối

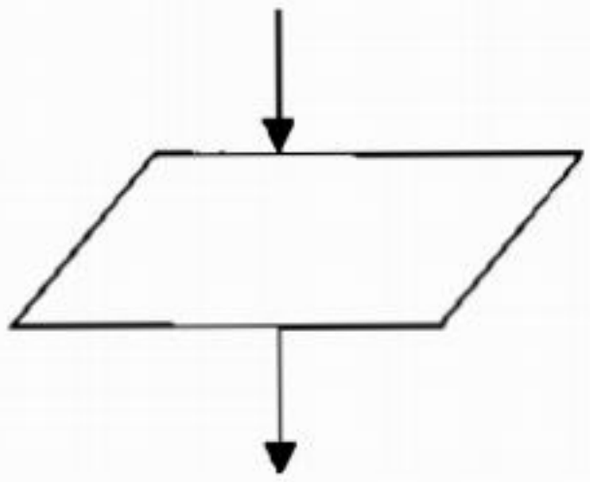
Cung định hướng:

Các khối thành phần trong sơ đồ khối phải được nối với nhau bằng các cung định hướng để biểu thị thuật toán.

Khối hình elip: chứa biểu thị điểm bắt đầu và kết thúc thuật toán. *Gọi là khối bắt đầu/ kết thúc*. Khối bắt đầu chỉ có một cung đi ra, khối kết thúc chỉ có một cung đi vào.

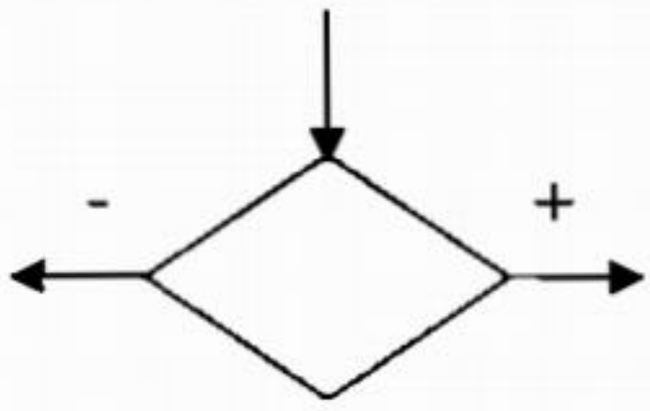


Khối hình bình hành: chứa các lệnh vào ra. *Gọi là khối vào/ra.*



Khối hình chữ nhật: chứa một hay nhiều lệnh. *Gọi là khối lệnh.*

Khối hình thoi: chứa một điều kiện (biểu thức logic) có giá trị đúng hoặc sai và luôn luôn có hai đường dẫn ra tương ứng với từng trường hợp của điều kiện. *Khối này còn gọi là khối rẽ nhánh.*

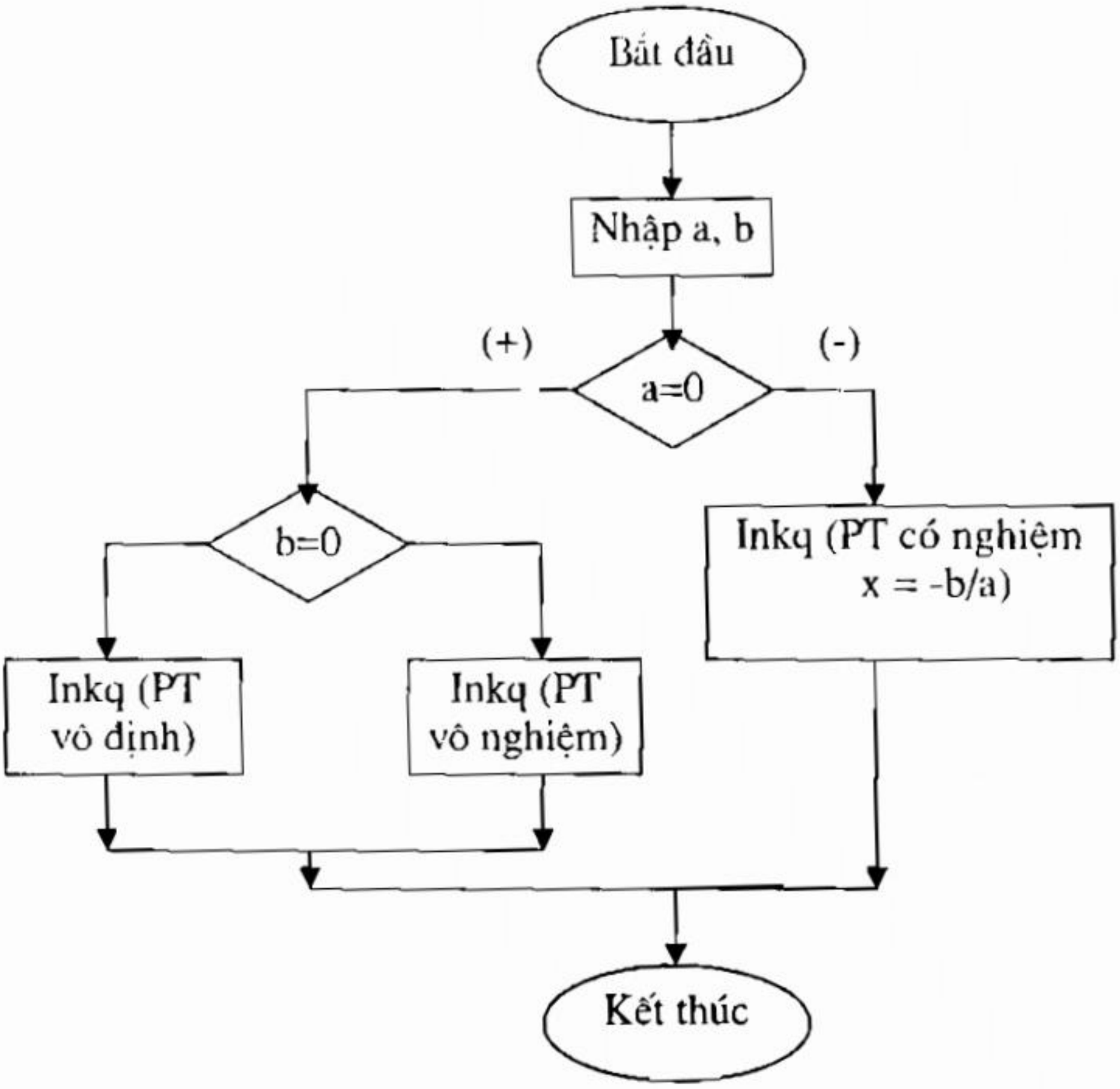


http://tieulun.hopto.org

Vì bản chất của khối vào/ra cũng chứa các lệnh nên nếu không sợ nhầm lẫn thì ta có thể không cần sử dụng thêm ký pháp hình bình hành và thay vào đó ta sẽ sử dụng ký pháp hình chữ nhật (khối lệnh) để ghi các lệnh nhập dữ liệu và in kết quả.

Việc sử dụng sơ đồ khối để mô tả thuật toán sẽ được minh hoạ kỹ trong mục 2.2. Tuy nhiên, chúng ta có thể tham khảo trước một ví dụ sau đây:

Ví dụ 1.5: Bằng sơ đồ khối, mô tả thuật toán giải phương trình: ax + b = 0.



1.3. Sử dụng phương pháp giả lập mã (pseudo code)

Một cách khác để mô tả thuật toán là dùng giả lập mã. Phương pháp này sử dụng ngôn ngữ tự nhiên nhưng gần gũi với ngôn ngữ lập trình bậc cao để diễn tả thuật toán. Ngôn ngữ tự nhiên trong giả lập mã dễ chuyển đổi sang ngôn ngữ lập trình bậc cao, do vậy người ta gọi phương pháp này là pseudo code - nghĩa là giả lập mã lệnh.

Một thuật toán được mô tả bằng giả lập mã có thể viết dạng như sau:

http://tieulun.hopto.org

Algorithm (thuật toán): Tên thuật toán.

Function (chức năng): Chức năng, nhiệm vụ của thuật toán.

Input (vào): Dữ liệu vào của thuật toán.

Output (Ra): Dữ liệu ra của thuật toán.

Format (khuôn dạng): Quy tắc, quy ước hay cú pháp biểu diễn các đại lượng output được tìm từ các đại lượng input.

Method (phương pháp): Phần chính mô tả thuật toán gồm các lệnh của thuật toán để giải quyết bài toán. Các lệnh được viết bằng ngôn ngữ tự nhiên nhưng gần gũi với mã lệnh của ngôn ngữ lập trình bậc cao (thường thì giống với ngôn ngữ Pascal). Các dòng lệnh này thường được đánh số thứ tự.

End (Tên thuật toán) *(kết thúc thuật toán)*.

Ví dụ 1.6: Xây dựng thuật toán tính tập Z là hợp của 2 tập X và Y.

Định nghĩa: Cho 2 tập hợp X và Y các phần tử cùng bản chất, hợp của 2 tập X và Y (ký hiệu là $X \cup Y$) là một tập hợp bao gồm các phần tử có trong X hoặc Y.

Trên cơ sở định nghĩa trên, ta xây dựng thuật toán tính tập Z bằng giả lập mã như sau:

Algorithm Union.
Function Hợp của 2 tập hợp.
Input Tập X, tập Y.
Output $Z = X \cup Y = \{ a \mid a \in X \vee a \in Y \}$
Format $Z = X \cup Y$
Method
1. Khởi tạo tập Z bằng rỗng;
2. Với mỗi phần tử a thuộc tập X thì làm như sau:
3. Thêm a vào tập Z;
4. Với mỗi phần tử a thuộc tập Y thì làm như sau:
5. Nếu kiểm tra thấy a không thuộc tập Z thì:
6. Thêm a vào tập Z;
7. Xác nhận tính xong tập Z;

End Union.

http://tieulun.hopto.org

Lưu ý: Không có quy tắc bắt buộc nào để diễn đạt các lệnh trong giả lập mã. Ban đầu tiếp cận với thuật toán, người ta thường dùng hoàn toàn tiếng Việt để biểu thị các lệnh. Tuy nhiên, khi đã biết về các cấu trúc điều khiển trong thuật toán, nhất là khi đã biết về một ngôn ngữ lập trình cụ thể thì ta nên sử dụng kết hợp các cú pháp câu lệnh bằng tiếng Anh của ngôn ngữ lập trình đó với các ký hiệu riêng sao cho (**nhất quán, logic và tự nhiên**) để biểu đạt giả lập mã. Thuật toán trên có thể sẽ không khó hiểu nếu ta viết lại như sau:

Algorithm Union.
Function Hợp của 2 tập hợp.
Input Tập X, tập Y.
Output $Z = X \cup Y = \{ a \mid a \in X \vee a \in Y \}$
Format $Z = X \cup Y$
Method

1. Z = { };
2. For each a in X do
3. Add a to Z;
4. For each a in Y do
5. If not (a in Z) then
6. Add a to Z;
7. Return Z;

End Union.

Nhận xét: Trong ba hình thức diễn tả thuật toán (liệt kê từng bước, sơ đồ khối và giả mã), không thể khẳng định chắc chắn hình thức nào tốt nhất. Mỗi cách mô tả thuật toán đều có những ưu điểm và nhược điểm nhất định. Điều đó tuỳ thuộc vào từng bài toán cụ thể và phần nào tuỳ thuộc vào thói quen, ý thích, cách nhìn của từng người.

Nếu có một thuật toán khá rõ ràng và ta mong muốn sự mô tả nó có thể dễ chuyển đổi sang ngôn ngữ lập trình, ta nên sẽ chọn hình thức giả mã. Ngược lại, nếu một thuật toán phức tạp, chủ yếu chỉ nêu lên được chiến lược giải quyết bài toán và việc cài đặt thuật toán trên ngôn ngữ lập trình còn phải trải qua nhiều mức để cụ thể hoá hơn nữa thì ta nên lựa chọn sơ đồ khối hoặc liệt kê từng bước để chỉ ra đường đi, hoạt động của thuật toán.

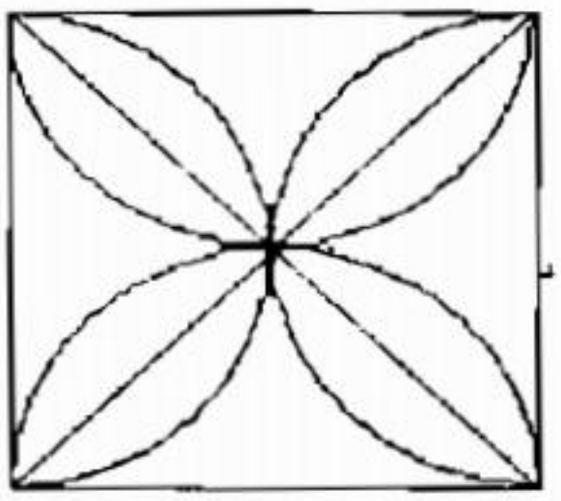
2. Các cấu trúc điều khiển trong thuật toán

2.1. Cấu trúc tuần tự

Cấu trúc tuần tự là cấu trúc mà trong đó các lệnh được thực hiện theo đúng thứ tự nó được chỉ ra trong thuật toán.

http://tieulun.hopto.org

Ví dụ 1.7: Cho một hình vuông cạnh bằng a. Người ta vẽ bốn nửa hình tròn nhận 4 cạnh hình vuông làm đường kính. Phần chung của các cặp đường tròn tạo thành một bông hoa 4 cánh. Xây dựng thuật toán tính diện tích của bông hoa đó?



Bước 1: Nhập cạnh a.
*Bước 2: Snt := (1/2)*pi*(a/2)*(a/2);*
{ diện tích nửa hình tròn}
*Stg := (1/2)*a*(a/2);*
{diện tích tam giác nội tiếp trong hình tròn}
S2nc := Snt-Stg;
{ diện tích 2 nửa cánh hoa}
*Sbh := 4*S2nc;*
{ diện tích bông hoa}
Bước 3: In kq (Diện tích bông hoa là: Sbh);
Bước 4: Kết thúc.

Lưu ý: Dấu hoa thị * dùng để ký hiệu phép nhân (phù hợp với cú pháp của các ngôn ngữ lập trình).

2.2. Cấu trúc rẽ nhánh

Cấu trúc rẽ nhánh là cấu trúc trong đó một hay một nhóm lệnh được thực hiện hay không được thực hiện là phụ thuộc vào giá trị đúng hay sai của một điều kiện nào đó.

Ví dụ 1.8: Xây dựng thuật toán giải phương trình $ax^2 + bx + c = 0$ với a,b,c là các số thực cho trước.

Bước 1: Nhập các số thực a,b,c;
Bước 2: Nếu a<>0 thì làm Bước 8;
Bước 3: Nếu b<>0 thì làm Bước 7;
Bước 4: Nếu c<>0 thì làm Bước 6;
Bước 5: - Inkq (PT VĐ) { a=b=c=0}
- Thực hiện Bước 12;

http://tieulun.hopto.org

Bước 6: - Inkq (PT VN) {a=b=0, c<>0}

- Thực hiện Bước 12;

Bước 7: - Inkq (PT có nghiệm x=-c/b);

{ a=0, b<>0}

- Thực hiện Bước12;

*Bước 8: Tính d := b*b-4*a*c;*

Bước 9: Nếu d<0 thì thực hiện Bước 5;

Bước 10: Nếu d=0 thì

- InKq

*(PT có nghiệm kép x = -b/(2*a))*

- Thực hiện Bước 12;

Bước 11: - Inkq

(Phương trình có 2 nghiệm phân biệt:

*x1=(-b+sqrt(d))/(2*a);*

*x2=(-b-sqrt(d))/(2*a);)*

Bước12: Kết thúc;

Nhận xét:

Có thể có nhiều hơn một thuật toán để giải một bài toán. Và, với một hình thức diễn tả thuật toán thì cũng có thể có nhiều cách thể hiện thuật toán đó.

Việc xây dựng thuật toán bằng phương pháp liệt kê từng bước có 2 cách thể hiện sau đây:

Cách thứ nhất là thể hiện theo phong cách lập trình chuyển điều khiển, tựa Basic, nghĩa là tận dụng các bước nhảy, bước chuyển điều khiển theo kiểu nếu gặp khó khăn thì "nợ ở bước tương lai".

Trong ví dụ 1.8, ta thấy ở bước 2: khi a <> 0 ta có một phương trình bậc 2 thực sự và việc giải nó được khất "nợ", nghĩa là chuyển điều khiển đến bước 8. Trong quá trình hình thành thuật toán, khi làm đến bước 2 ta sẽ không thể biết trước là bước tương lai của việc chuyển điều khiển là bước 8. Cho nên trong thực tế, tại đây ta tạm thời viết là:

Bước 2: Nếu a <> 0 thì làm Bước "nợ 1"

Bước 3: Nếu b <> 0 thì làm Bước "nợ 2"

...

Đến khi giải quyết hết những trường hợp phủ định bên dưới, ta mới biết chính xác "nợ 1" là bước 8 và "nợ 2" là bước 7.

Khi giải quyết những trường hợp phủ định bên dưới ta phải luôn luôn thấm nhuần ý nghĩa của sự *ngầm định* và tính chất của cấu trúc *tuần tự* của

http://tieulun.hopto.org

thuật toán. Chẳng hạn nếu bước 2 không thực hiện bước 8 thì theo cấu trúc tuần từ thuật toán sẽ tiến hành thực hiện bước 3 và ở bước 3 phải ngầm định là a = 0. Tương tự tại bước 3 nếu không thực hiện bước 7 thì theo cấu trúc tuần tự thuật toán sẽ thực hiện bước 5 và ở bước 5 luôn phải hiểu "ngầm định" rằng a = 0 và b = 0...

Phong cách hình thành thuật toán kiểu như thế này giúp cho việc gỡ rối vấn đề theo từng bước và có thể giải quyết được nhiều cấu trúc cồng kềnh khi phải biện luận nhiều khả năng xuất hiện trong bài toán.

Cách thứ hai là thể hiện theo phong cách của lập trình có cấu trúc, tựa Pascal. Trong cách này người ta tận dụng tối đa tính cấu trúc của loại lệnh đang được sử dụng. Cách này được ưa dùng hơn trong sơ đồ khối và không thích hợp trong liệt kê từng bước vì nó có thể làm cho một bước sẽ diễn ra rất cồng kềnh. "Thể hiện 1" dưới đây thể hiện cách "ứ 2 giải phương trình bậc 2 theo phong cách này một cách triệt để.

Nếu một sự biện luận khá phức tạp trong một bước nào đó mà bị phân rã thành các bước nhỏ hơn thì thực sự nó lại chuyển về phong cách gần Basic, nghĩa là trong trường hợp này là "lai" cả 2 phong cách.

Thể hiện 1:

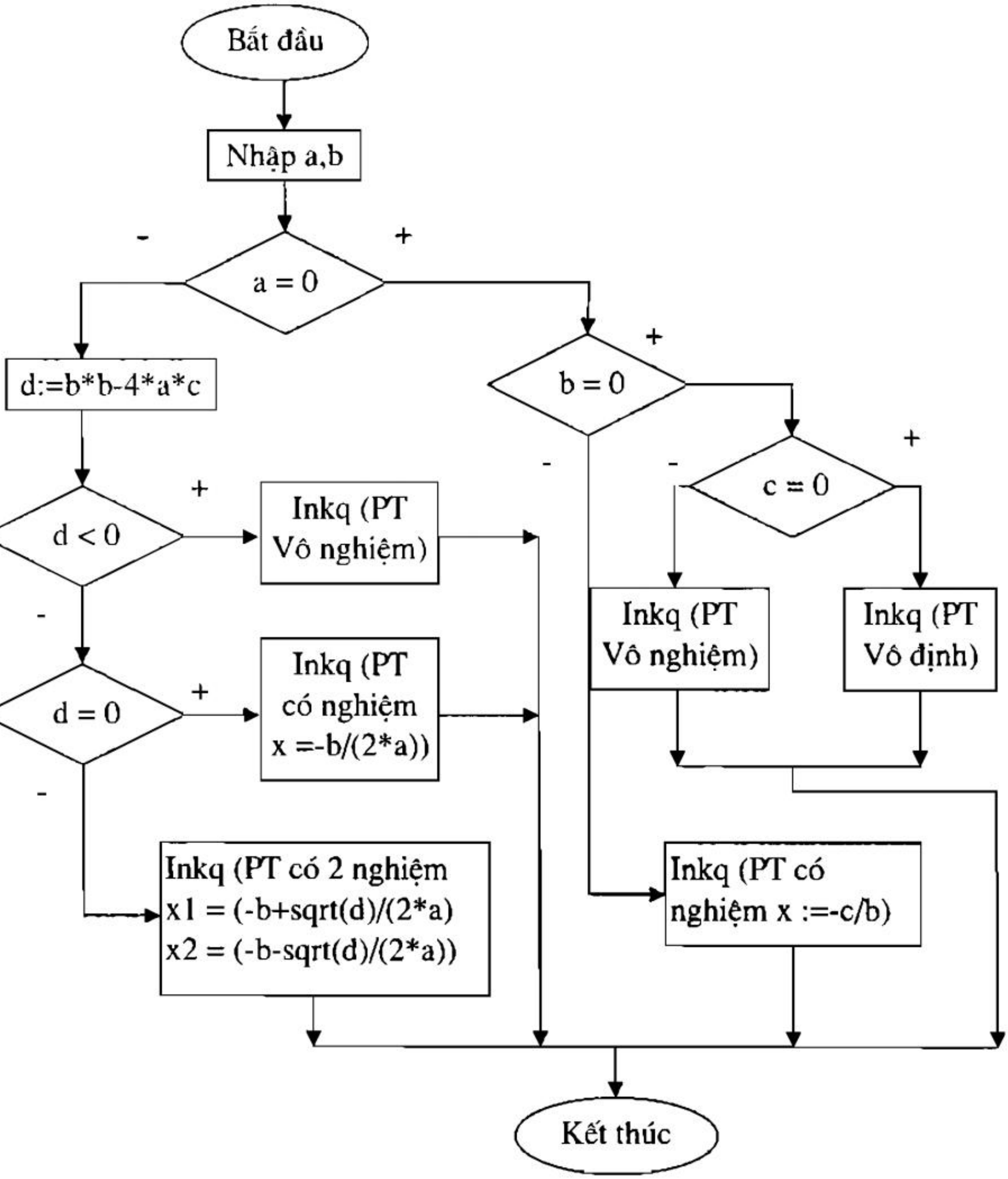
```
Bước 1: Nhập các số thực a,b,c;
Bước 2: Nếu a=0 thì
        Nếu b=0 thì
          Nếu c=0 thì
            Inkq
(Phương trình trình có vô số nghiệm)
          Không thì
            Inkq (Phương trình vô nghiệm)
        Không thì
          Inkq (Phương trình có nghiệm x = -c / b)
      Không thì làm các lệnh sau đây:
        Bước 2a: Tính d := b*b-4*a*c;
        Bước 2b: Nếu d<0 thì
              Inkq (Phương trình vô nghiệm)
            Không thì
            Nếu d=0 thì
              InKq (Có nghiệm kép x= -b/(2*a))
            Không thì
```


http://tieulun.hopto.org

Inkq (Phương trình có 2 nghiệm phân biệt:
*x1=(-b+sqrt(d))/(2*a);*
*x2=(-b-sqrt(d))/(2*a);)*
Bước 3: Kết thúc;

Ba bước này có thể diễn đạt bằng sơ đồ khối như thể hiện 2 sau đây:

Thể hiện 2:



http://tieulun.hopto.org

Trong quy ước diễn đạt thuật toán bằng hình thức liệt kê từng bước, khi có nhiều hơn một lệnh cùng chịu sự phụ thuộc vào một điều kiện trong một cấu trúc điều khiển (rẽ nhánh hoặc lặp) người ta cho phép dùng thêm khái niệm câu lệnh ghép có dạng: Bắtđầu <nhóm lệnh> Kếtthúc.

2.3. Cấu trúc lặp

Cấu trúc lặp là cấu trúc trong đó cho phép một hay một nhóm lệnh được lặp đi lặp lại một số hữu hạn lần.

Có một điều kiện dùng để điều khiển vòng lặp nghĩa là nó cho phép tiếp tục hay dừng quá trình lặp tùy thuộc vào giá trị đúng hay sai của nó, tất nhiên để đảm bảo tính dừng của vòng lặp thì bản thân điều kiện này có thể bị thay đổi giá trị dưới tác động của các lệnh trong vòng lặp đó.

Gần với đa số ngôn ngữ lập trình, cấu trúc lặp còn chia thành 2 loại:

Lặp với số lần biết trước.

Lặp với số lần không biết trước: gồm 2 loại là kiểm tra điều kiện trước và kiểm tra điều kiện sau. Điều kiện, ở đây dùng để điều khiển vòng lặp để quyết định cho nhóm lệnh được thực hiện lặp tiếp hay dừng lại, phụ thuộc vào giá trị đúng hay sai của điều kiện đó.

Ví dụ 1.9: Xây dựng thuật toán tính $T = a^n$ với mọi a và n là các số nguyên khác 0 cho trước. Trong ví dụ này giả thiết abs(k) là hàm cho giá trị tuyệt đối của số k.

Mô tả thuật toán bằng phương pháp liệt kê theo từng bước

Bước 1: Nhập a,n;

Bước 2: Nếu (a=0) hoặc (n=0) thì quay về Bước 1;

Bước 3: Khởi tạo T:=1; i:=0;

*Bước 4: i:=i+1; T:=T*a;*

Bước 5: Nếu i < abs(n) thì thực hiện Bước 4;

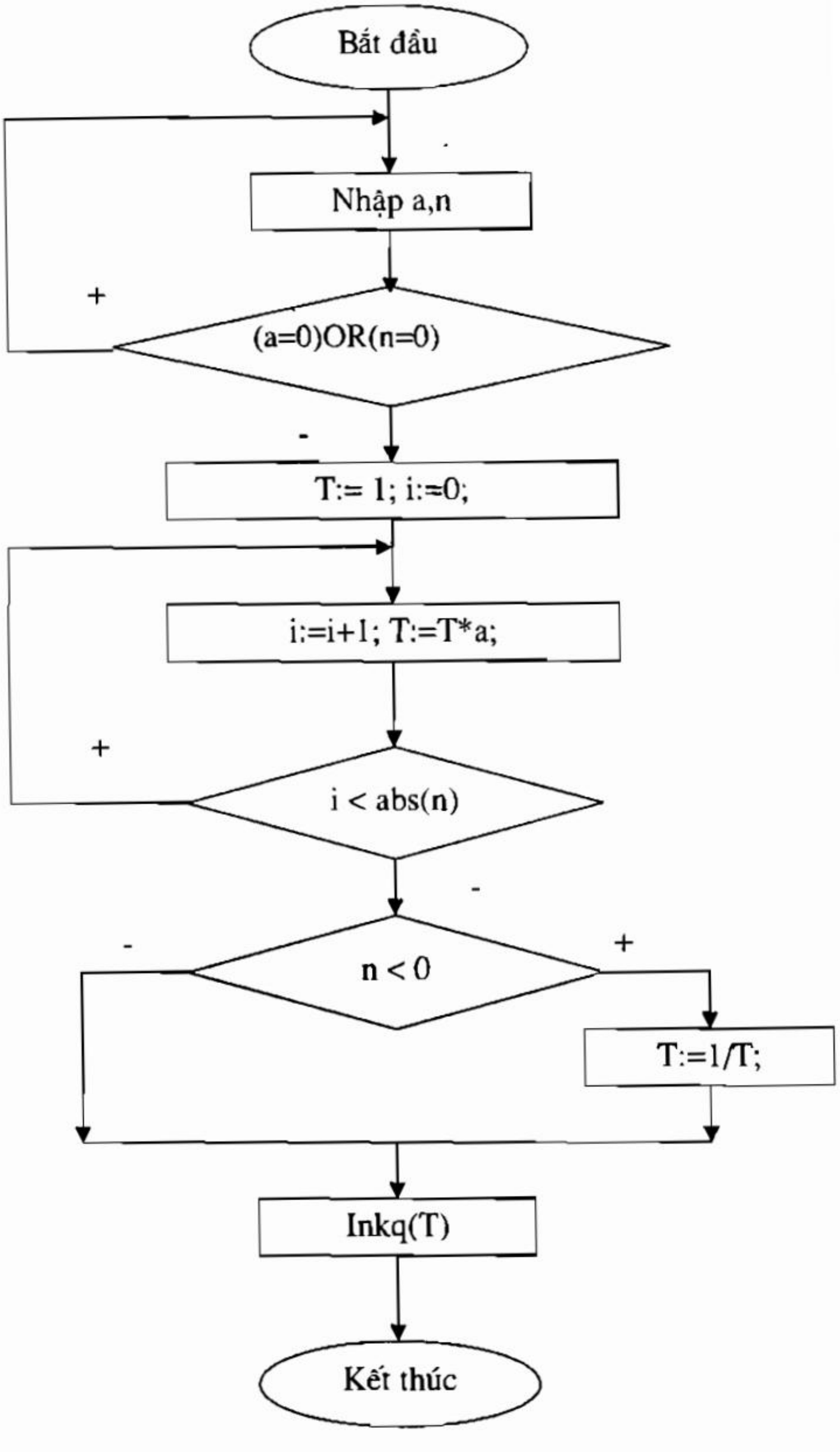
Bước 6: Nếu n<0 thì T:=1/T;

Bước 7: Inkq(T);

Bước 8: Kết thúc;

http://tieulun.hopto.org

Mô tả thuật toán bằng phương pháp sơ đồ khối



http://tieulun.hopto.org

Mô tả thuật toán bằng phương pháp giả lập mã

Algorithm HamMu.
Function Tính a mũ n.
Input số nguyên a ≠ 0, số nguyên n ≠ 0
Output T = a*a*...* a (có n thừa số a)
Format $T = a^n$
Method
1. Khởi tạo T=1;
2. For i=1 to n thực hiện lệnh
3. T=T*a;
4. If n < 0 Then gán lại
5. T = 1/T;
6. Return T;
End HamMu.

Ví dụ 1.10: Xây dựng thuật toán tìm ước số chung lớn nhất của hai số nguyên a và b cho trước.

Phương pháp trừ liên tiếp

Mô tả thuật toán: Lặp quá trình sau đây khi a còn khác b: thay số lớn hơn bằng hiệu của nó trừ đi số bé hơn. Kết thúc quá trình lặp thì 2 số bằng nhau và một trong chúng là ước số chung lớn nhất của 2 số ban đầu.

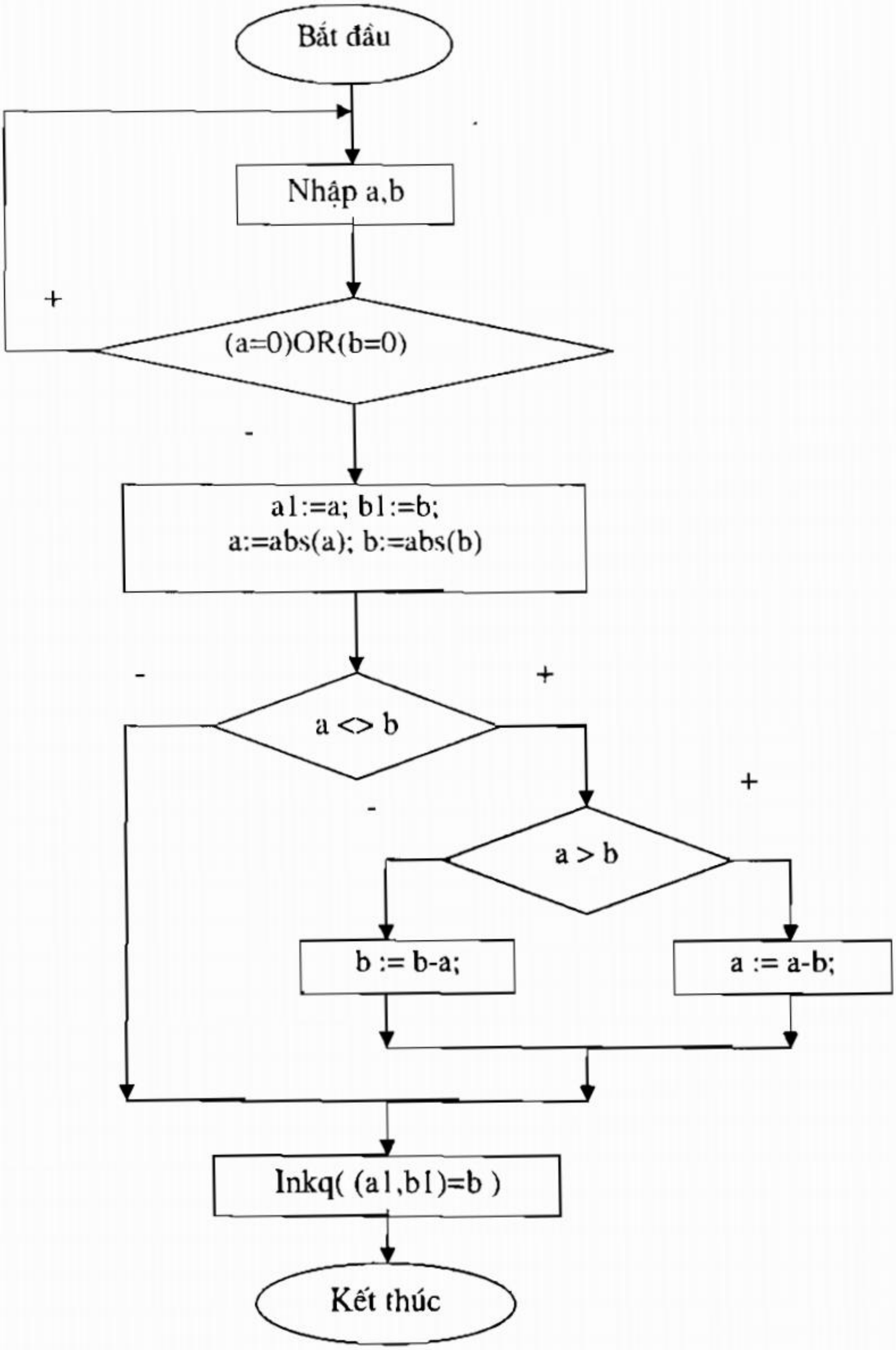
Mô tả thuật toán bằng cách liệt kê theo từng bước

Bước 1: Nhập a,b
Bước 2: Đặt a1=a; b1=b; a=abs(a); b=abs(b);
Bước 3: Nếu a=b thì làm Bước 6;
Bước 4: Nếu a>b thì a:=a-b
Không thì b:=b-a;
Bước 5: Quay về Bước 3;
Bước 6: Inkq (USCLN của a1 và b1 là b);
Bước 7: Kết thúc.

http://tieulun.hopto.org

Mô tả thuật toán bằng sơ đồ khối

Ta thống nhất sử dụng ký pháp <> để chỉ phép so sánh khác nhau ≠



http://tieulun.hopto.org

Mô tả thuật toán bằng hình thức giả lập mã

Algorithm UCLN.

Function Tính ước số chung lớn nhất của 2 số nguyên.

Input số nguyên a <> 0, số nguyên b <> 0

Output U = UCLN (a,b)

Format U = UCLN (a,b)

Method

1. While a <> b làm các bước sau:
2. a) If a > b Then thay a bằng b;
3. b) Else If b > a Then thay b bằng a;
4. U = b;
5. Return U ;

End UCLN.

Phương pháp chia liên tiếp

Mô tả thuật toán: Tính r là số dư của số nguyên a chia cho số nguyên b. Lặp quá trình sau đây khi dư r còn khác 0: thay a bằng b, thay b bằng r và tính lại dư r. Kết thúc quá trình lặp thì b là ước số chung lớn nhất của 2 số ban đầu.

Mô tả thuật toán bằng cách liệt kê theo từng bước

Trong mô tả thuật toán ta sẽ sử dụng phép toán mod để chỉ phép toán lấy phần dư. Ví dụ a mod b sẽ cho kết quả là số dư của phép chia số nguyên a cho số nguyên b.

Bước 1: Nhập a,b

Bước 2: Đặt a1=a; b1=b; a=abs(a); b=abs(b);

Bước 3: Tính r = a mod b;

Bước 4: Nếu r=0 thì làm Bước 7;

Bước 5: a:=b; b:=r; r:= a mod b;

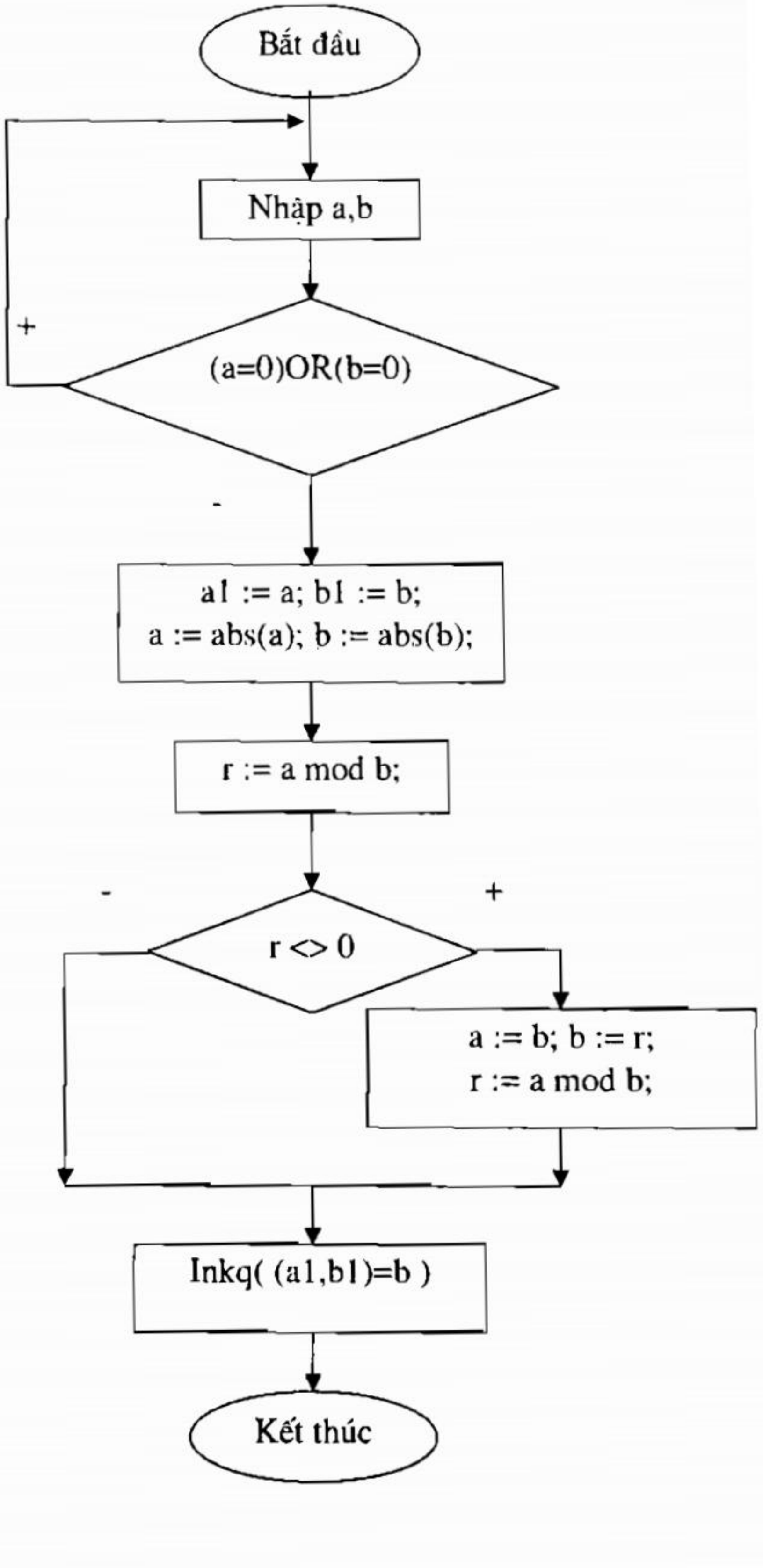
Bước 6: Quay về Bước 4;

Bước 7: Inkq (USCLN của a1 và b1 là b);

Bước 8: Kết thúc.

http://tieulun.hopto.org

Mô tả thuật toán bằng sơ đồ khối



http://tieulun.hopto.org

Mô tả thuật toán bằng hình thức giả lập mã

Algorithm UCLN.
Function Tính ước số chung lớn nhất của 2 số nguyên.
Input số nguyên a ≠ 0, số nguyên b ≠ 0
Output U = UCLN (a,b)
Format U = UCLN (a,b)
Method
1. Tính r là số dư của a chia cho b;
2. While <> 0 làm các bước sau:
3. a) Thay a bằng b;
4. b) Thay b bằng r;
5. c) Tính lại r là số dư của a chia cho b;
6. Gán U = b;
7. Return U;
End UCLN.

3. Câu hỏi và bài tập

Bài 1.4

Định nghĩa các cấu trúc điều khiển trong thuật toán? Cho ví dụ minh hoạ?

Bài 1.5

Để tìm ước số chung lớn nhất của 2 số a và b tương ứng có giá trị 100 và 10 thì thuật toán trừ liên tiếp thực hiện các lệnh lặp bao nhiêu lần? Thuật toán chia liên tiếp thực hiện các lệnh lặp bao nhiêu lần?

Bài 1.6

Xây dựng thuật toán tính P = n! và trình bày:
a) Bằng hình thức liệt kê từng bước.
b) Bằng sơ đồ khối.
c) Bằng giả lập mã.

Bài 1.7

Dùng sơ đồ khối để trình bày thuật toán tính tổng các lập phương của n số tự nhiên đầu tiên.

Bài 1.8

Dùng hình thức thức liệt kê từng bước để trình bày thuật toán in ra các số có 2 chữ số và số đó chia hết cho 3.

http://tieulun.hopto.org

III. GIỚI THIỆU CÁC NGÔN NGỮ LẬP TRÌNH

1. Khái niệm

Ngôn ngữ lập trình là ngôn ngữ dùng để viết chương trình máy tính. Chương trình máy tính chính là một thể hiện cụ thể của thuật toán bằng một dạng thức quy định bằng một ngôn ngữ.

2. Phân loại (ngôn ngữ bậc thấp, ngôn ngữ bậc cao)

2.1. Ngôn ngữ máy

Ngôn ngữ máy (còn gọi là chương trình mã máy) là tập hợp các lệnh thực tại điều khiển các mạch logic vi lập trình trong đơn vị xử lý trung tâm. Chương trình viết bằng ngôn ngữ máy về mặt logic chỉ bao gồm các ký hiệu 0 và 1 (được thể hiện về mặt vật lý là 2 trạng thái và chỉ có 2 trạng thái khác nhau của các vi mạch điện tử), tạo thành các mã nhị phân để biểu thị vi lệnh của chip xử lý trung tâm.

2.2. Hợp ngữ (Asembly)

Hợp ngữ là một ngôn ngữ lập trình (viết trong môi trường – hay chương trình dịch – Asembler) bao gồm một tập các lệnh sơ cấp (đơn giản và nhỏ nhất): INPUT, PRINT, LOAD, ADD, SUB, MUL, DIV, MOVE.... Chú ý rằng mọi mệnh lệnh đều có thể phân tích thành một tập các lệnh sơ cấp. Nói cách khác tập hợp hữu hạn các lệnh sơ cấp trong hợp ngữ phủ hết các chỉ thị khác nhau để giải một bài toán tin học.

Ví dụ (đọc thêm)

Chương trình hợp ngữ tính e =(a+b)*(c+d):

```
INPUT a: {nạp a từ bàn phím}
INPUT b;
INPUT c;
INPUT d;
LOAD a; {nạp a vào thanh ghi tổng A}
ADD b; { cộng A với b, kết quả đặt vào A}
MOVE e; {ghi A vào e}
LOAD c;
ADD d;
```


http://tieulun.hopto.org

```
MUL e;
MOVE e;
PRINT e;
HALT
a
b
c
d
e
END
```

2.3. Chương trình dịch

Chương trình dịch là một chương trình đặc biệt cho phép dịch một chương trình viết bằng một ngôn ngữ lập trình sang ngôn ngữ máy.

Chương trình dịch thực hiện các chức năng sau đây:

Duyệt chương trình nguồn để phát hiện và thông báo các lỗi cú pháp do người lập trình viết sai.

Dịch chương trình nguồn thành chương trình viết bằng ngôn ngữ máy.

Thực hiện chương trình mã máy đó.

2.4. Ngôn ngữ lập trình bậc thấp và bậc cao

Có thể viết trực tiếp một chương trình viết bằng ngôn ngữ máy, nhưng điều đó rất khó khăn và hay sinh lỗi. Các nhà lập trình dựa trên nguyên lý của máy tính được điều khiển bằng chương trình để sáng tạo ra các ngôn ngữ lập trình gần gũi với ngôn ngữ con người, phù hợp với tâm lý và tư duy con người để diễn đạt các thuật toán một cách trong sáng và tự nhiên.

Việc sáng tạo ra hợp ngữ là bước đầu thay thế ngôn ngữ máy bằng các lệnh dễ hiểu với người dùng. Tuy nhiên các nhà tin học tiếp tục sáng tạo ra các ngôn ngữ lập trình mới gần gũi với ngôn ngữ tự nhiên hơn. Để dễ dàng phân biệt người ta quy ước gọi:

Ngôn ngữ máy và hợp ngữ là ngôn ngữ lập trình bậc thấp.

Ngôn ngữ lập trình còn lại là ngôn ngữ lập trình bậc cao.

3. Tóm tắt về các lớp ngôn ngữ (đọc thêm)

Có thể phân loại các lớp ngôn ngữ theo lịch sử phát triển của các ngôn ngữ lập trình:

http://tieulun.hopto.org

3.1. Thế hệ 1

Lập trình ở mức mã máy (từ năm 1950) mà điển hình là hợp ngữ (ngôn ngữ lập trình gần ngôn ngữ máy nhất).

3.2. Thế hệ 2

Câu lệnh của hợp ngữ được gộp lại bằng các câu lệnh có tính cấu trúc của ngôn ngữ thế hệ 2 (từ cuối năm 1950 đến hết năm 1960).

Bao gồm FORTRAN, COBOL, ALGOL và cao hơn một chút là BASIC.

FORTRAN: thích hợp để giải các bài toán khoa học, kỹ thuật. Thiếu tính cấu trúc, nghèo về kiểu dữ liệu. Bản gốc là FORTRAN-66, bản chuẩn ANSI là FORTRAN-77.

COBOL: thích hợp để giải các bài toán xử lý dữ liệu thương mại, tuy nhiên vẫn không gọn gàng trong cấu trúc lệnh.

ALGOL: cho phép lập trình với thủ tục và định nghĩa kiểu dữ liệu mới. Có thể lập trình cấp phát bộ nhớ động, đệ quy - là tiên phong của ngôn ngữ thế hệ 3.

3.3. Thế hệ 3

Ngôn ngữ lập trình hiện đại, có cấu trúc. Định nghĩa cấu trúc dữ liệu và thủ tục mạnh. Chia thành 3 lớp:

Ngôn ngữ cấp cao vạn năng:

Bao gồm các ngôn ngữ PL/1, PASCAL, Modula-2, C và Ada.

PL/1: gần như đã xếp vào ngôn ngữ thế hệ 2,5. Nó là ngôn ngữ phổ rộng đúng nghĩa đầu tiên. Được xây dựng với một phạm vi rộng các tính năng làm cho nó dùng được với nhiều lĩnh vực ứng dụng khác nhau: hỗ trợ các ứng dụng kỹ nghệ khoa học và kinh doanh; đặc tả các cấu trúc dữ liệu phức tạp, đa nhiệm, vào/ra phức tạp, xử lý danh sách và nhiều tính năng khác.

PASCAL là ngôn ngữ lập trình hiện đại được phát triển đầu năm 1970 để lập trình có cấu trúc và phát triển phần mềm, thích hợp cho các ứng dụng khoa học kỹ thuật, lập trình hệ thống (PASCAL còn gọi là FORTRAN của những năm 1980 và là con cháu trực tiếp của ALGOL).

Modula-2 theo một khía cạnh nhận định, là sự trưởng thành của PASCAL, cài đặt các tính năng thiết kế như che dấu thông tin, trừu tượng và kiểm tra kiểu mạnh với các cấu trúc điều khiển để hỗ trợ cho đệ quy và tương tranh; hạn chế cho các ứng dụng công nghiệp.

http://tieulun.hopto.org

C ban đầu được phát triển dành cho người cài đặt hệ điều hành. Hệ điều hành UNIX được cài đặt trong C. Từ đó C được sử dụng mạnh mẽ để tạo ra các sản phẩm phần mềm, các ứng dụng nhúng và phần mềm hệ thống. Giống như các ngôn ngữ khác, trong phạm trù này, C hỗ trợ cho các cấu trúc dữ liệu phức tạp, có các đặc trưng kiểu hợp lý, cho phép dùng con trỏ và có một tập phong phú các toán tử và thao tác dữ liệu. Bên cạnh đó, C làm cho người lập trình "gần gũi với máy hơn" bằng cách đưa vào các tính năng tựa hợp ngữ.

Ada sử dụng trong quốc phòng, tựa PASCAL về cấu trúc nhưng mạnh và phức tạp hơn, có khả năng đa nhiệm, điều khiển giải ngắt, đồng bộ hoá nhiệm vụ và truyền thông.

Các ngôn ngữ hướng đối tượng:

Cài đặt các mô hình phân tích thiết kế. Điển hình là C++, Smalltalk và Eiffel. Đều được sử dụng để hỗ trợ trực tiếp cho các định nghĩa lớp, kế thừa, bao bọc và truyền thông báo.

Các ngôn ngữ chuyên dụng:

Là các ngôn ngữ được đặc trưng bởi các dạng cú pháp bất thường được thiết kế cho các ứng dụng riêng. Ví dụ: LISP, PROLOG, APL, FORTH...

LISP: thích hợp cho thao tác xử lý danh sách hay gặp trong các bài toán tổ hợp. Dùng cho trí tuệ nhân tạo, chứng minh định lý, tìm kiếm theo cây và các hoạt động giải quyết vấn đề khác. Ngày nay dùng để phát triển các hệ chuyên gia.

PROLOG giống như LISP, đưa các chức năng hỗ trợ biểu diễn tri thức, sử dụng cấu trúc dữ liệu duy nhất gọi là term.

APL thao tác trên mảng và vector, giải quyết các bài toán toán học.

FORTH thiết kế phát triển phần mềm bộ xử lý.

3.4. Thế hệ 4 (4GL - 4 generation language)

Bao gồm:

- Ngôn ngữ hỏi.
- Bộ sinh chương trình.
- Các 4GL khác.

4. Thông dịch và biên dịch (đọc thêm)

Chương trình dịch có 2 cách thức làm việc: thông dịch và biên dịch.

http://tieulun.hopto.org

4.1. Thông dịch

Theo cách này, hành động do câu lệnh của ngôn ngữ quy định được thực hiện trực tiếp. Thông thường với mỗi một hành động đều có tương ứng một chương trình con - viết trong ngôn ngữ máy - để thực hiện nó. Vậy việc thông dịch chương trình được thực hiện bằng cách gọi các chương trình con theo một dãy thích hợp.

Chính xác hơn, bộ thông dịch là một chương trình (dịch) thực hiện lặp các thao tác sau:

1. Lấy một câu lệnh kế tiếp.
2. Xác định hành động cần thực hiện.
3. Thực hiện hành động.

Dãy này rất giống với mẫu hình hoạt động của một máy tính truyền thống, tức là:

1. Nhặt lệnh tiếp (lệnh có địa chỉ được xác định bởi con trỏ lệnh).
2. Tăng con trỏ lệnh (tức là đặt con trỏ vào lệnh kế tiếp cần nhặt).
3. Giải mã lệnh.
4. Thực hiện lệnh.

Sự tương tự này chỉ ra rằng việc thông dịch có thể được xem như sự mô phỏng trên máy tính chủ, cho một máy chuyên dụng có ngôn ngữ máy là một ngôn ngữ cấp cao.

4.2. Biên dịch

Theo cách thức này, chương trình viết trong ngôn ngữ cấp cao được biên dịch thành *bản* theo ngôn ngữ máy tương đương trước khi thực hiện. Việc biên dịch này tiến hành theo nhiều bước. Các chương trình con trước hết được biên dịch thành mã hợp ngữ; mã hợp ngữ được biên dịch thành "mã máy tính khả tái định vị"; các đơn vị trong mã khả tái định vị được móc nối với nhau để tạo thành mã máy khả tái định vị duy nhất; cuối cùng, toàn bộ chương trình được nạp vào bộ nhớ dưới dạng mã máy thực hiện được. Bộ biên dịch dùng trong mỗi bước có các tên riêng: trình biên dịch, trình hợp dịch, trình móc nối (hoặc trình soạn thảo liên kết), trình nạp tương ứng.

5. Câu hỏi và bài tập

Bài 1.9

Các khái niệm:

- Ngôn ngữ máy?

http://tieulun.hopto.org

- Hợp ngữ?

- Chương trình dịch?

- Ngôn ngữ lập trình bậc thấp và ngôn ngữ lập trình bậc cao? Pascal thuộc loại ngôn ngữ lập trình nào?

Bài 1.10

"Học thuật toán nghĩa là học Pascal" - khẳng định đó đúng hay sai? Thuật toán dùng cho mọi ngôn ngữ lập trình hay một ngôn ngữ lập trình dùng cho mọi thuật toán?

http://tieulun.hopto.org

Chương 2

CÁC KIỂU DỮ LIỆU CƠ BẢN VÀ CÁC CẤU TRÚC ĐIỀU KHIỂN

I. CÁC THAO TÁC CƠ BẢN TRONG TURBO PASCAL

1. Các file của Turbo Pascal

Để chạy chương trình Turbo Pascal (Turbo Pascal For Ms-DOS) có thể lựa chọn các file sau đây:

Cấu hình tối thiểu: gồm 2 file TURBO.EXE và TURBO.TPL

Cấu hình vừa đủ: gồm các file TURBO.* và các file để viết các chương trình đồ hoạ: GRAPH.TPU, *.BGI, *.CHR.

• Mỗi một kiểu màn hình yêu cầu một file BGI tương ứng để điều khiển. Đa số màn hình hiện nay sử dụng một file EGAVGA.BGI là đủ. Tuy nhiên nếu viết một phần mềm đồ họa có thể đáp ứng được đối với đa số họ màn hình và cho phép chương trình đồ họa tự nhận ra kiểu màn hình (auto DETECT) thì nên có đủ các file BGI.

• Mỗi một file *.CHR cho phép quản lý một loại font trong chương trình đồ họa.

2. Vào/ra Turbo Pascal

Nguyên tắc chạy thực hiện TP (Turbo Pascal) theo 2 bước:

Bước 1: Chuyển vào thư mục chứa chương trình TP.

Bước 2: Chạy thực hiện chương trình TURBO.EXE.

• Đường dẫn ở bước 1 là tuỳ theo quy định của người cài đặt hay copy chương trình.

• Có nhiều cách thực hiện Turbo.exe: từ dấu nhắc của Ms-dos, từ Nc, từ windows...

http://tieulun.hopto.org

- Alt-X: để thoát khỏi chương trình.

3. Tóm tắt các phim tắt để quản lý file

- Mở file mới: Alt - F, N (File \ new).
- Ghi file: Alt - F, S.
- Ghi file với tên mới: Alt - F. A.
- Đóng file: Alt-F3.
- Mở file đã có: F3.
- Chuyển tới một cửa sổ chương trình: F6 hoặc Alt-stt cửa sổ.

4. Tóm tắt các thao tác soạn thảo

- Đánh dấu khối: Shift và định vị bằng các phím mũi tên, phím home, end.
- Sao chép: Đánh dấu khối, nhấn Ctrl-C, định vị đến vị trí đích, Ctrl-V.
- Di chuyển: Đánh dấu khối, nhấn Ctrl-X, định vị đến vị trí đích, Ctrl-V.
- Lùi, tiến một khối: Ctrl- K- I (tiến), Ctrl- K- U (lùi).
- Khôi phục thao tác: Ctrl-shift - Back space.
- Sử dụng bộ nhớ tạm Clipboard: để sao chép và di chuyển:

Để sao chép: Đánh dấu khối, nhấn Ctrl-Insert, định vị đến vị trí đích. Shift-Insert

Để di chuyển: Đánh dấu khối, nhấn Ctrl-Insert, xóa bằng phím Delete, định vị đến vị trí đích, Shift-Insert.

- Theo dõi chương trình theo từng bước (debug): F4, F7, F8.

5. Các bước thực hành

Bước 1: Chạy TP.

Bước 2: Đặt tên file (F2).

Bước 3: Soạn thảo chương trình.

Bước 4: Dịch chương trình (F9) để sửa lỗi cú pháp. Dịch nhiều lần đến khi hết lỗi cú pháp.

Bước 5: Chạy chương trình (Ctrl-F9). Có thể chạy nhiều lần để kiểm tra với nhiều dữ liệu kiểm thử khác nhau.

Bước 6: Nếu muốn kết thúc thực hành thì thực hiện Bước 8, nếu muốn thực hiện bài mới thì: Mở file mới (F3).

Bước 7: Quay về bước 2.

Bước 8: Thoát khỏi TP (Alt-X).

http://tieulun.hopto.org

Chú ý:

Nếu chương trình có thuật toán phức tạp, tổ chức nhiều biến và ở bước 4 (dịch chương trình) có nhiều lỗi, đặc biệt là khi bị sai sót về cài đặt thuật toán ở bước 5 (chạy chương trình) thì có thể kiểm tra sự thay đổi giá trị của các biến trong từng bước chạy chương trình bằng phương pháp "step by step" kết hợp với việc theo dõi giá trị của một số biến quan trọng. Sử dụng Ctrl-F7, F7, F8.

II. GIỚI THIỆU CHUNG VỀ NGÔN NGỮ LẬP TRÌNH PASCAL

1. Các phần tử cơ sở của ngôn ngữ lập trình Pascal

1.1. Bộ ký tự

Ngôn ngữ Pascal sử dụng tập ký tự, bao gồm:

- Các chữ cái.
- Các chữ số.
- Các dấu phép toán.
- Và các ký tự khác

1.2. Từ khoá, từ chuẩn và tên gọi

Các ký tự tạo thành các từ trong ngôn ngữ. Các từ này tạo thành các **tên gọi.** Từ khoá và từ chuẩn là các tên gọi có sẵn trong ngôn ngữ.

Từ khoá xác định một ý nghĩa duy nhất trong chương trình và người lập trình không thay đổi được ý nghĩa có sẵn của nó.

Các từ khoá bao gồm:

Các từ khoá dùng để khai báo: USES, CONST, TYPE, VAR, BEGIN, END, PROGRAM, ARRAY, RECORD...

Các từ khoá dùng để biểu thị câu lệnh có cấu trúc: IF THEN, CASE OF, WHILE DO, FOR DO, REPEAT UNTIL....

Và các từ khoá khác...

Từ chuẩn cũng xác định một ý nghĩa có sẵn nào đó trong chương trình nhưng nó có thể bị người lập trình định nghĩa lại nó để mang một ý nghĩa mới.

Các từ chuẩn thường là tên gọi các thủ tục và hàm có sẵn, tên gọi các kiểu dữ liệu đơn giản chuẩn (có sẵn). Ví dụ: integer, real, byte,...., sqrt, abs,...

Tuy nhiên khác với từ khóa, người ta có thể định nghĩa lại các từ chuẩn này.

http://tieulun.hopto.org

Tất cả các **tên gọi khác** do người dùng tự quy định, ví dụ như tên hằng, tên biến, tên kiểu, tên thủ tục, tên hàm tự xây dựng và nói chung có độ dài không vượt quá 64 ký tự.

Chú ý tất cả các tên gọi nói chung (từ khoá và từ chuẩn, tên biến, tên kiểu...) không được chứa dấu cách bên trong vì chúng là các từ của ngôn ngữ.

1.3. Câu lệnh

Câu lệnh trong ngôn ngữ lập trình là sự kết hợp các từ khoá, từ chuẩn, các tên gọi nói chung và các toán tử theo một quy tắc cú pháp nhất định để tạo thành chỉ thị của chương trình.

Các câu lệnh ngăn cách nhau bởi dấu chấm phẩy.

Trong Pascal có 2 loại câu lệnh:

Câu lệnh đơn giản gồm: lệnh gán, lệnh vào/ra và lệnh gọi thực hiện các hàm và thủ tục.

Câu lệnh có cấu trúc: để biểu thị câu lệnh phức tạp điều khiển rẽ nhánh hoặc lặp trong chương trình.

2. Kiểu dữ liệu, hằng, biến, biểu thức và câu lệnh gán

2.1. Kiểu dữ liệu

Kiểu dữ liệu (data type) là loại dữ liệu mà một biến có khả năng nhận được trong chương trình.

Giả sử ta phải viết một chương trình Pascal để tính giá trị hàm $f(x) = 1/x^2$ tại giá trị nào đó (nhập từ bàn phím) của biến x. Trong toán học, hàm f(x) này xác định trên tập số thực khác 0 và nhận giá trị trên tập số thực dương. Trong tin học, ở góc độ lập trình, ta nói rằng biến x và hàm f chỉ được phép nhận loại dữ liệu là các số thực, hay nói cách khác, x và f có kiểu dữ liệu là số thực. Ngôn ngữ Pascal dùng từ chuẩn real để chỉ kiểu dữ liệu là số thực. Pascal nói riêng và các ngôn ngữ lập trình nói chung không có các từ chuẩn khác để quy định kiểu số thực khác 0 và kiểu số thực dương. Người lập trình phải viết lệnh để kiểm tra x khác 0 (do đó, f sẽ dương khi tính theo x).

Có thể khẳng định rằng, trong chương trình: tất cả các đại lượng mà có giá trị đều phải gắn với một kiểu dữ liệu nhất định để quy định loại giá trị mà đại lượng đó được phép nhận.

http://tieulun.hopto.org

Các kiểu dữ liệu trong ngôn ngữ lập trình không phải là chỉ có kiểu số mà còn có các kiểu dữ liệu cụ thể để chỉ tập các giá trị là: số thực, số nguyên, xâu ký tự, các giá trị logic (đúng và sai),... Ta sẽ được tìm hiểu đầy đủ hơn về kiểu dữ liệu trong các phần sau.

2.2. Hằng

Hằng là một đại lượng không thay đổi trong suốt quá trình chương trình thực hiện.

Hằng được khai báo dạng:

CONST tên_hằng = giá_trị_hằng;

Ví dụ:

```
CONST
   maxN =100;          {hằng kiểu dữ liệu là số nguyên}
   Pi = 3.1416;        {hằng kiểu dữ liệu là số thực}
   Sperator = ';';     {hằng kiểu dữ liệu là ký tự}
   title = 'PASCAL;    {hằng kiểu dữ liệu là xâu ký tự}
   ON = TRUE;          {hằng kiểu dữ liệu là logich - boolean}
```

2.3. Biến

Biến là một đại lượng có thể thay đổi giá trị trong chương trình.

Biến đặc trưng cho một ô nhớ trong bộ nhớ, bao gồm:

- Tên biến: đại diện cho địa chỉ ô nhớ.
- Giá trị của biến: là nội dung của ô nhớ đó.

Biến có vai trò:

- Chứa dữ liệu vào.
- Chứa dữ liệu ra.
- Chứa các giá trị trung gian trong quá trình tính toán.

Tên biến do người dùng quy định (một trường hợp của tên gọi).

Biến luôn luôn phải được khai báo trước khi sử dụng và trong câu lệnh khai báo biến phải chỉ rõ kiểu dữ liệu của biến.

Câu lệnh khai báo biến có dạng:

VAR tên_biến : Tên_kiểu_dữ liệu;

Ví dụ:

```
VAR n,m : integer; {m và n là các biến nguyên}
    x,y : real;    {x và y là các biến thực}
```


http://tieulun.hopto.org

2.4. Biểu thức

Biểu thức là sự kết hợp các hằng và các biến bởi các phép toán thích hợp trên kiểu dữ liệu tương ứng với các hằng và biến đó. Giá trị thu được của biểu thức thuộc một kiểu dữ liệu xác định.

Tùy thuộc vào kiểu dữ liệu (kiểu giá trị) của biểu thức mà ta có biểu thức nguyên, biểu thức thực, biểu thức ký tự, biểu thức xâu ký tự, biểu thức logic...

Bản thân một hằng hoặc một biến cũng được coi là một biểu thức đơn giản.

Ví dụ: Sau đây là các biểu thức:

```
123
45.6
'P'
'HELLO'
Pi*r*r
(x+y)/(2*m);
FALSE
```

Những lỗi có dạng "*Type Mismatch*" thường xảy ra khi sử dụng sai kiểu dữ liệu. Ví dụ nếu ta khai báo a,b,c: integer thì lệnh c:= a/b sẽ báo lỗi vì vế phải là một biểu thức real, không thể gán cho biến integer ở vế trái.

Hiện tượng giá trị của biểu thức vượt ra ngoài phạm vi biểu diễn cũng thường xảy ra. Trong trường hợp này chương trình biên dịch không báo lỗi nên có thể dẫn tới hiện tượng sai sót về kết quả. Ví dụ nếu ta khai báo x: integer và y: longint; xét các lệnh sau đây:

```
x := 200;
y := 1000*x;
```

sẽ cho giá trị sai của y.

Trong trường hợp này ta ngộ nhận cho rằng điều này vô lý vì ta đã khai báo y là longint? Thực ra thì một biểu thức tính toán với một biến integer sẽ cho một giá trị integer nên 1000*y sẽ bị tràn biểu diễn đối với số nguyên và kết quả "bị cắt" đó sẽ gán cho y. Cách khắc phục trong trường hợp này là khai báo x cũng là longint. Nói tóm lại, các giá trị của biểu thức tính được cần phải được ước lượng để xét xem nó có bị vượt quá phạm vi biểu diễn đối với một kiểu dữ liệu nào đó đang quan tâm hay không.

http://tieulun.hopto.org

2.5. Câu lệnh gán

Câu lệnh gán có dạng:

Biến := Biểu_thức_ cùng_kiểu_dữ_liệu ;

Ví dụ:

```
x:= 5.6;
y:= m/n + x*x;
```

Câu lệnh gán được thực thành 2 bước, phải hết sức chú ý điều này:

Bước 1: Tính giá trị biểu thức về phải

Bước 2: Gán giá trị biểu thức vừa tính cho biến ở vế trái.

Ta hãy xem xét hoạt động của câu lệnh gán trong hai ví dụ sau đây:

Ví dụ 1:

Xét dãy các lệnh sau	Bước 1		Bước 2	
	Vế trái	Vế phải	Vế trái	Vế phải
x := 10;		10	10	10
x := x + 1;	10	11	11	11
x := 2*x + x;	11	33	33	33

Ví dụ 2:

Xét dãy các lệnh sau	Biến vế trái
a := 5; b := 7;	a = 5 b = 7

c := a;
a := b;
b := c;
	c = 5
	a = 7
	b = 5

3. Các lệnh vào/ra

3.1. In thông tin lên màn hình: lệnh write, writeln

Lệnh writeln có hai cách sử dụng như sau:

WRITELN;

WRITE[LN] (biểu thức);

http://tieulun.hopto.org

Nếu trong biểu thức có biểu thức là hằng xâu ký tự thì phải viết hằng xâu ký tự đó trong nháy đơn. Để tiện phân biệt hằng xâu ký tự với các biểu thức còn lại, ta gọi biểu thức hằng là xâu ký tự **là thông báo.**

Nếu biểu thức không phải là thông báo thì phải viết nó giữa hai dấu phẩy. Trường hợp riêng: nếu biểu thức viết ở đầu lệnh writeln thì chỉ cần một dấu phẩy đằng sau nó, nếu biểu thức đứng ở cuối lệnh writeln thì chỉ cần một dấu phẩy đằng trước nó.

Trong lệnh writeln có thể chỉ có thông báo hoặc chỉ có biểu thức.

Ví dụ:

Giả sử có x:=5; y:=7; i:=3. Xét các lệnh sau đây:

Lệnh	Kết quả trên màn hình
Writeln('Hello !');	Hello
Writeln(x);	5
Writeln('x= ',x,', y=',y);	x=5, y=7
Writeln('x va y = ',x,' va ',y);	x va y = 5 va 7
Writeln('x,' la x, ',y,' la y, => x + y = ', x+y,'.');	5 la x, 7 la y, => x + y = 12.
Writeln('a[',i,'] = ',x+y);	a[3]= 12
Writeln('USCLN: (',x,',',y,') = ',1);	USCLN(x,y)=1

Chú ý:

Lệnh write chỉ khác lệnh writeln ở chỗ: lệnh write đặt con trỏ màn hình ở cuối dòng hiện tại sau khi in ra thông báo và biểu thức, lệnh writeln đặt con trỏ màn hình xuống đầu dòng dưới.

Đặc biệt lệnh writeln; chỉ có tác dụng xuống dòng.

Ví dụ:

Hai lệnh write('See you.'); Writeln('How are you today?'); có thể thay bằng một lệnh: Writeln('See you. How are you today?');

3.2. Đọc dữ liệu từ bàn phím - lệnh readln

Lệnh đọc dữ liệu từ bàn phím có dạng:

Readln(biến 1, biến 2,..., biến n);

http://tieulun.hopto.org

Tác dụng: Cho phép người chạy chương trình nhập các giá trị cho các biến chỉ ra trong lệnh readln. Nói chung, các giá trị được nhập từ bàn phím ngăn cách nhau bởi một dấu cách (phím space). Kết thúc nhập dữ liệu bằng phím Enter.

Đặc biệt lệnh **Readln** không đọc bất kỳ giá trị nào từ bàn phím cho một biến nào đó mà chỉ có tác dụng tạm dừng thực hiện chương trình (chức năng Pause). Lệnh readln thực hiện xong nếu người dùng bấm phím Enter. Thường đặt lệnh Readln cuối chương trình để xem kết quả trước khi trở về chương trình nguồn hoặc đặt xen kẽ ở chỗ nào đó trong chương trình để tạm dừng xem từng trang kết quả trên màn hình trong quá trình chạy chương trình.

4. Cấu trúc cơ bản của một chương trình Pascal

4.1. Cấu trúc cơ bản của một chương trình Pascal

Một chương trình Pascal gồm 3 phần:

Phần tiêu đề: khai báo tên chương trình (được phép bỏ qua phần này).

PROGRAM Tên_chương_trình;

Phần khai báo (phần này có thể không có nếu không có mục khai báo nào):

Có các loại khai báo sau đây:

+ Dẫn hướng biên dịch. Ví dụ sau đây là dẫn hướng tắt kiểm tra dữ liệu vào:

  {$I+}

+ Thư viện chương trình, khai báo như sau:

  USES Tên_thư_viện;

+ Hằng, khai báo như sau:

  CONST tên_hằng = giá_trị;

+ Kiểu, khai báo như sau:

  TYPE tên_kiểu = Mô_tả_kiểu;

+ Biến, như ta đã biết, khai báo như sau:

  VAR tên_biến: Kiểu_dữ_liệu;

+ Khai báo và xây dựng các chương trình con (ta sẽ xem xét vấn đề này sau).

Phần thân chương trình (bắt buộc phải có):

```
BEGIN
  Các_lệnh_giải_quyết_bài_toán;
END.
```


http://tieulun.hopto.org

Chú ý:

Trước phần thân chương trình có thể có phần khai báo và xây dựng các chương trình con, ta sẽ xem xét sau.

Dẫn hướng biên dịch và một số khai báo khác (như khai báo biến) thậm chí có thể viết tại các vị trí khác nhau trước khi sử dụng nó, miễn sao không được viết trong phần thân chương trình.

Các lệnh trong chương trình được ngăn cách bởi dấu chấm phẩy ";" và không nhất thiết mỗi lệnh một dòng. Trong chương trình có thể viết thêm các lời chú thích tuỳ ý miễn sao các chú thích này viết giữa hai dấu ngoặc nhọn. hoặc giữa hai cặp ký pháp (* và *). Ví dụ như:

{nội dung chú thích thứ nhất}

(* nội dung chú thích thứ hai *)

Lưu ý là không có dấu cách giữa hai ký tự trong ký pháp (* và trong ký pháp *).

4.2. Ví dụ về chương trình Pascal đơn giản

Ví dụ 2.1:

Viết chương trình in lên màn hình dòng chữ GLAD TO MEET YOU.

VIDU1.PAS

```
BEGIN
  Writeln('GLAD TO MEET YOU !');
  Readln;
END.
```

Trong ví dụ 2.1, chương trình được bỏ qua phần tiêu đề. Chương trình không sử dụng biến nào do đó nó không có phần khai báo. Chương trình chỉ gồm hai lệnh: lệnh thứ nhất in lên màn hình dòng chữ GLAD TO MEET YOU, lệnh thứ hai tạm dừng chương trình để người chạy chương trình được quan sát dòng chữ đó cho đến khi muốn quay trở về màn hình viết chương trình thì nhấn phím bất kỳ.

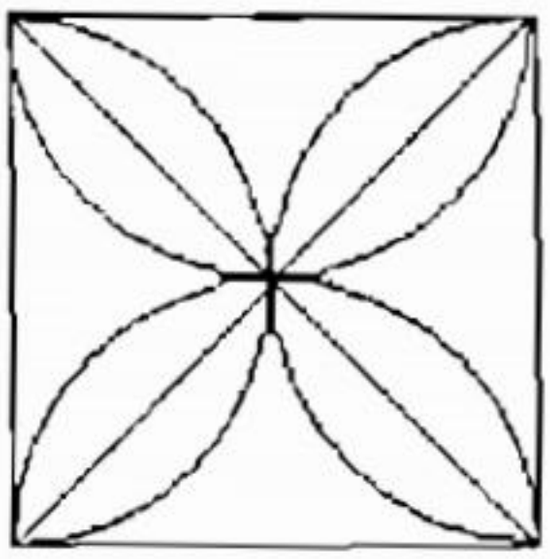
Ví dụ 2.2:

Tính diện tích bông hoa trong ví dụ 1.7 của chương 1

http://tieulun.hopto.org

VIDU2.PAS

```
const Pi=3.1416;
Var  a : real;
    snt,stg,s2nc,sbh : real;
Begin
    Write('Nhap canh hinh vuong: ');
    Readln(a);
    Snt := (1/2)*pi*(a/2)*(a/2);
    Stg := (1/2)*a*(a/2);
    S2nc := Snt-Stg;
    Sbh := 4*S2nc;
    Writeln(' Dien tich bong hoa la: ', Sbh :10:3);
    Readln;
End.
```



Giá trị của biến sbh trong lệnh writeln nói trên được in ra màn hình ở dạng có quy cách: "sbh : 10 : 3" nghĩa là giá trị của sbh được in lên màn hình là một số thực chiếm 10 vị trí: phần sau dấu chấm thập phân chiếm 3 vị trí, một vị trí cho dấu chấm thập phân và phần nguyên chiếm 6 vị trí.

Chương trình ở ví dụ 2.2 là một minh hoạ dễ hiểu rằng: chương trình giải một bài toán, nói chung gồm ba phần:

- Nhập dữ liệu.
- Tính toán, giải quyết bài toán trên dữ liệu vào.
- In kết quả.

Trong chương trình trên: Hai lệnh đầu để nhập dữ liệu là cạnh a của hình vuông. Bốn lệnh gán tiếp theo thể hiện cách tính diện tích bông hoa (sbh) bằng 4 lần diện tích hai nửa cánh hoa (s2nc), trong đó diện tích hai nửa cánh hoa thì bằng diện tích nửa hình tròn (snt) trừ đi diện tích của tam giác (stg); dễ dàng

http://tieulun.hopto.org

tính được diện tích nửa hình tròn khi biết đường kính bằng a, còn diện tích tam giác được biết nhờ đường cao bằng a/2 và cạnh đáy bằng a. Hai lệnh cuối cùng của chương trình được xem như hai lệnh in lên màn hình diện tích bông hoa sau quá trình tính toán và dừng lại xem kết quả đó.

Lưu ý: Ta có thể không cần khai báo hằng số Pi trong chương trình trên vì tên gọi Pi đã có sẵn trong ngôn ngữ.

Xét quá trình tính toán trong chương trình trên ta có:

sbh = 4*((1/2)*pi*(a/2)*(a/2) - (1/2)*a*(a/2))

sbh = a*a*(pi/2 - 1)

Vì vậy chương trình trên có thể viết ngắn gọn như sau:

```
Var  a, sbh : real;
Begin
   Write('Nhap canh hinh vuong: '); Readln(a);
   sbh := a*a*(pi/2 - 1 )
   Writeln(' Dien tich bong hoa la: ',Sbh :10:3);
   Readln;
End.
```

Thậm chí có thể viết ngắn gọn hơn nữa:

```
Var  a : real;
Begin
   Write('Nhap canh hinh vuong: '); Readln(a);
   Writeln(' Dien tich bong hoa la: ',a*a*(pi/2 - 1 ):10:3);
   Readln;
End.
```

Khi ta cần diễn tả rõ thuật toán trong chương trình (ở ví dụ trên là cách tính diện tích bông hoa) thì không nên lược bớt các lệnh thể hiện điều đó. Vậy trong trường hợp này ta không nên chọn cách viết các chương trình quá ngắn gọn. Tuy nhiên, khi gặp bài toán khó hơn, các tính toán như ở ví dụ trên chỉ là một phần nhỏ so với các phần chính khác của thuật toán. Trong trường hợp này, ta có thể rút gọn quá trình tính toán để diễn tả những ý đồ lớn hơn và là ý chính trong thuật toán đang xét.

http://tieulun.hopto.org

5. Câu hỏi và bài tập

Bài 2.1

Chỉ ra các cách chạy chương trình TURBO.EXE trong các trường hợp khác nhau?

Bài 2.2

Phân biệt lỗi về cú pháp và lỗi về thuật toán khi viết chương trình?

Bài 2.3

Biến đặc trưng cho ô nhớ của máy tính như thế nào? Biến có những vai trò gì?

Bài 2.4

Viết chương trình nhập từ bàn phím số thực r. Tính và in lên màn hình chu vi, diện tích hình tròn bán kính r.

Bài 2.5

Viết chương trình, nhập từ bàn phím độ dài 3 cạnh a, b, c của một tam giác. Giả thiết dữ liệu nhập vào là đúng đắn.

In lên màn hình các giá trị sau đây:

- Chu vi, diện tích,
- Độ dài 3 đường cao,
- Độ dài 3 đường trung tuyến,
- Độ dài 3 đường phân giác,
- Độ dài bán kính đường tròn nội tiếp và ngoại tiếp.

Cho trước các công thức:

- Nửa chu vi: P=(a+b+c)/2;
- Diện tích: S= sqrt (p(p-a)(p-b)(p-c))
- Đường cao tại cạnh a: $h_a = 2S/a$
- Trung tuyến tại a: $m_a = (1/2)\ sqrt\ (2b^2 + 2c^2 - a^2)$
- Phân giác tại a: $l_a = (2/(b+c))\ sqrt(bcP(P-a))$
- Bán kính đường tròn nội tiếp R = abc/(4S); ngoại tiếp r = S/P;

III. CÁC KIỂU DỮ LIỆU CƠ BẢN

1. Đặt vấn đề

Như ta đã biết: kiểu dữ liệu quy định loại dữ liệu mà một biến có khả năng nhận được. Một cách đầy đủ hơn ta thấy rằng:

- Có các kiểu dữ liệu khác nhau.

http://tieulun.hopto.org

- Mỗi một kiểu dữ liệu quy định loại thao tác (các phép toán) tác động lên dữ liệu thuộc kiểu đó.

- Mỗi một kiểu dữ liệu được chương trình dịch xác định có một phạm vi biểu diễn cụ thể.

Tiếp theo đây ta sẽ tìm hiểu về các vấn đề nói trên.

2. Tổng quan phân loại

Kiểu dữ liệu đơn giản chuẩn là các kiểu dữ liệu đơn giản có sẵn trong ngôn ngữ. Các *kiểu dữ liệu có cấu trúc* và các *kiểu dữ liệu do người lập trình định nghĩa* đều xây dựng trên cơ sở các kiểu dữ liệu đơn giản chuẩn.

- *Các kiểu dữ liệu đơn giản chuẩn:*

Integer, Byte, Real, Char, Boolean.

- *Các kiểu dữ liệu có cấu trúc:*

Kiểu khoảng con, đoạn con và tập hợp.

Array.

String (nửa cấu trúc).

Record.

File.

- *Kiểu dữ liệu động:* Liên quan đến con trỏ, bản ghi tự trỏ, các danh sách liên kết, các cấu trúc cây.

3. Các kiểu dữ liệu đơn giản chuẩn

3.1. Integer và các kiểu mở rộng

3.1.1. Phạm vi biểu diễn

Kiểu	Phạm vi biểu diễn	Bộ nhớ cấp phát	Dạng biểu diễn
Shortint	-128..127	1 Bytes	Có dấu
Integer	-32.768..32.767	2 Bytes	Có dấu
Longint	-2.147.483.648... -2.147.463.847	4 Bytes	Có dấu
Byte	0..255	1 Bytes	Không dấu
Word	0..65535	2 Bytes	Không dấu

3.1.2. Các phép toán

Các phép toán đối với số nguyên +, -, *, MOD, DIV, các phép so sánh. Ngoài ra có thể có các phép toán Bit trên số nguyên.

http://tieulun.hopto.org

3.1.3. Các hàm và thủ tục

ODD, INC, DEC, RANDOM, ABS, SQR

+ Hàm ODD(n) là hàm kiểu Boolean, có giá trị TRUE hay FALSE tuỳ theo số nguyên n là lẻ hay chẵn. Việc kiểm tra tính chẵn lẻ của một số nguyên n có thể sử dụng phép toán MOD (chia lấy phần dư). Nếu xảy ra đẳng thức n mod 2 = 0 thì chứng tỏ hàm ODD(n) cho kết quả FALSE.

+ Hai thủ tục (lệnh) INC và DEC áp đụng trên tập có thứ tự, trong đó số nguyên cũng là một tập có thứ tự.

- INC(n); tương đương với lệnh n:= n+1;
- DEC(n); tương đương với lệnh n:= n-1;
- INC(n,a); tương đương với lệnh n:= n+a;
- DEC(n,a); tương đương với lệnh n:= n-a;

+ Hàm ABS(a) cho giá trị tuyệt đối của số a nói chung.

+ Hàm SQR(a) cho giá trị của a^2.

+ Hàm RANDOM(n) cho giá trị là một số nguyên k thoả mãn $1 \le k+1 \le n$.

3.2. Real và các kiểu mở rộng

3.2.1. Phạm vi biểu diễn

Kiểu	Phạm vi biểu diễn	Bộ nhớ cấp phát	Số chữ số có nghĩa
Real	2.9E-29 ..1.7E38	6 Bytes	11..12
Single	1.5E-45.. 3.4E38	4 Bytes	7..8
Double	5.0E-324..1.7E308	8 Bytes	15..16
Extended	3.4E-4932..1.1E4932	10 Bytes	19..20
Comp	-9.2E18..9.2E18	8 Bytes	19..20

Chú ý rằng các kiểu số thực mở rộng chỉ sử dụng được khi ta có hướng dẫn biên dịch trong trong chương trình là {N+} hoặc trong màn hình chương trình Turbo Pascal ta chọn từ menu Option lệnh Compiler, tiếp theo, trong vùng Numberic Proccessing, ta bật chọn trong ô 8087/80287. Một trong hai cách làm trên là để chương trình dịch sử dụng giá lập bộ đồng xử lý số học 8087.

3.2.2. Các phép toán

Các phép toán đối với số thực (đại diện là kiểu real) bao gồm các phép toán trên số nguyên và phép chia (/).

http://tieulun.hopto.org

Lưu ý, riêng các phép toán SHIFT, AND, OR, NOT, XOR, MOD, DIV chỉ áp dụng trên số nguyên.

3.2.3. Các hàm và thủ tục

Các hàm SQRT, ROUND, TRUNC, INT, FRAC, SIN, ARCTAN, COS, LN, EXP, ABS, SQR.

+ Hàm SQRT(r) cho giá trị là căn bậc của r.

+ Hàm ROUND(r) cho giá trị là số nguyên (kiểu longint) được làm tròn từ số thực r.

Ví dụ ROUND(234.689) sẽ cho kết quả là 235.

+ Hàm TRUNC(r) cho giá trị là phần nguyên (kiểu longint) của số thực r.

Ví dụ TRUNC(234.689) sẽ cho kết quả là 234.

+ Hàm INT(r) cho giá trị là phần nguyên (kiểu real) của số thực r.

Ví dụ ROUND(234.689) sẽ cho kết quả là 234.000.

+ Hàm FRAC(r) cho giá trị là phần thập phân (kiểu real) của số thực r.

Ví dụ FRAC(234.689) sẽ cho kết quả là 0.689.

3.3. Char

3.3.1. Phạm vi

Kiểu char (kiểu ký tự) chiếm 1 Byte.

Gồm 256 ký tự, tra cứu trong bảng mã ASCII mở rộng. Bảng mã ASCII (American Standard Code For Information Interchange - Bảng mã chuẩn của Mỹ) được sử dụng phổ biến. Bảng gồm 2 cột: cột số thứ tự và cột ký tự, trong đó số thứ tự của một ký tự chính là mã ASCII của ký tự đó trong bảng mã.

Cần phải ghi nhớ mã ASCII của một số ký tự:

Mã	Ký tự
13	ENTER
32	SPACE
48	0
49	1
...	...
57	9

Mã	Ký tự
65	A
66	B
...	...
90	Z

Mã	Ký tự
97	a
98	b
...	...
122	z

http://tieulun.hopto.org

3.3.2. Các phép toán

Phép gán, phép so sánh

Ví dụ: Phân biệt các lệnh sau đây:

Var m,n: char;

(1) m:= 'm';

(2) M:= 'm';

(3) m:= 'M';

Lệnh 1 và 2 có tác dụng giống nhau vì tên biến không phân biệt chữ thường và chữ hoa.

Lệnh 1 và 3 có tác dụng khác nhau vì giá trị ký tự chữ hoa khác giá trị ký tự chữ thường (mã ASCII khác nhau).

Trong lệnh 1, vế trái là tên biến, tuỳ ý muốn người lập trình quy định, còn vế phải là một biểu thức có giá trị thuộc kiểu Char (kiểu ký tự).

3.3.3. Các hàm trên kiểu char

CHR, ORD, UPCASE.

- PRED, SUCC (trên tập có thứ tự)

Ví dụ:

```
Var kt: char ;
    k : byte ;
kt := CHR (65); { kt='A'}
k  := ORD('A') ; { k= 65 }
```

Hệ quả:

Chữ cái in hoa thứ n (n≤ 26) là CHR(64+n).

Chữ cái in thường thứ n (n≤ 26) là CHR(96+n).

Chữ số p có giá trị số là ORD(p) – 48.

Chú ý: Ký tự # đứng trước một giá trị kiểu byte là cách viết một hằng ký tự, có tác dụng tương đương hàm CHAR. Ví dụ thay vì viết CHR(65), có thể viết là #65.

3.4. Boolean

3.4.1. Phạm vi biểu diễn

Kiểu boolean chiếm 1 Byte, chỉ gồm 2 giá trị [FALSE,TRUE].

http://tieulun.hopto.org

3.4.2. Các phép toán logic

Nếu a, b là 2 biến kiểu Boolean, ta có bảng các phép toán logic And, Or, Not sau đây.

Để tiện quan sát, giá trị True và False được ký hiệu trong bảng tương ứng là 1 và 0:

a	b	a and b	a or b	not a
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

- Phép gán và phép so sánh

Ví dụ:

```
Var ok, stop, good: Boolean;
ok:= (3>5) or (3<9);
writeln(ok); {Kết quả là TRUE}
writeln((3>5) and (3<9)); {Kết quả là FALSE}
good:= not ok;
Writeln(good); {Kết quả là FALSE;}
```

Trong lập trình, biến boolean thường được sử dụng để đánh dấu, ghi nhận một sự kiện nào với ý nghĩa sự kiện đó có xảy ra hay không xảy ra. Ví dụ "có" hay "không có" tên một học sinh nào đó trong một danh sách tên học sinh của một lớp. Ngoài ra, các biểu thức boolean phức tạp (tổ hợp từ nhiều biểu thức boolean) thường được tách ra nhờ sử dụng thêm một biến boolean. Ta sẽ thấy được vai trò của biến boolean trong các ví dụ và bài tập của giáo trình này.

4. Câu hỏi và bài tập

Bài 2.6

Khái niệm kiểu dữ liệu? Tổng quan phân loại?

Bài 2.7

Giả sử có các khai báo biến:

http://tieulun.hopto.org

```
var a,b: integer;
      r: real;
```

Khi đó, các lệnh sau đây sai vì sao?

```
a:= a/b;
writeln(r mod 2);
writeln(b:10:3);
```

Bài 2.8

Giải thích tác dụng của các lệnh sau đây:

```
Write(Char(65),'nh', #13, #10,'Binh');
Write(34>56);
```

Bài 2.9

Tính giá trị cho các biến (kiểu boolean) error, good, ok, stop trong các lệnh gán sau đây:

```
x:=9; m:=101;
error:= (x < 0) or (x >10);
good:= Not Error;
ok:= (m >= 10) and (m <=99);
stop:= good or ok;
```

IV. CÁC CẤU TRÚC ĐIỀU KHIỂN TRONG NGÔN NGỮ LẬP TRÌNH PASCAL

1. Cấu trúc tuần tự

Các lệnh viết liên tiếp nhau và theo thứ tự đó chương trình thực hiện tạo thành cấu trúc tuần tự.

Các chương trình trong ví dụ 2.1 và ví dụ 2.2 là minh hoạ chương trình chỉ gồm các lệnh đơn giản thực hiện theo cấu trúc tuần tự.

2. Cấu trúc rẽ nhánh

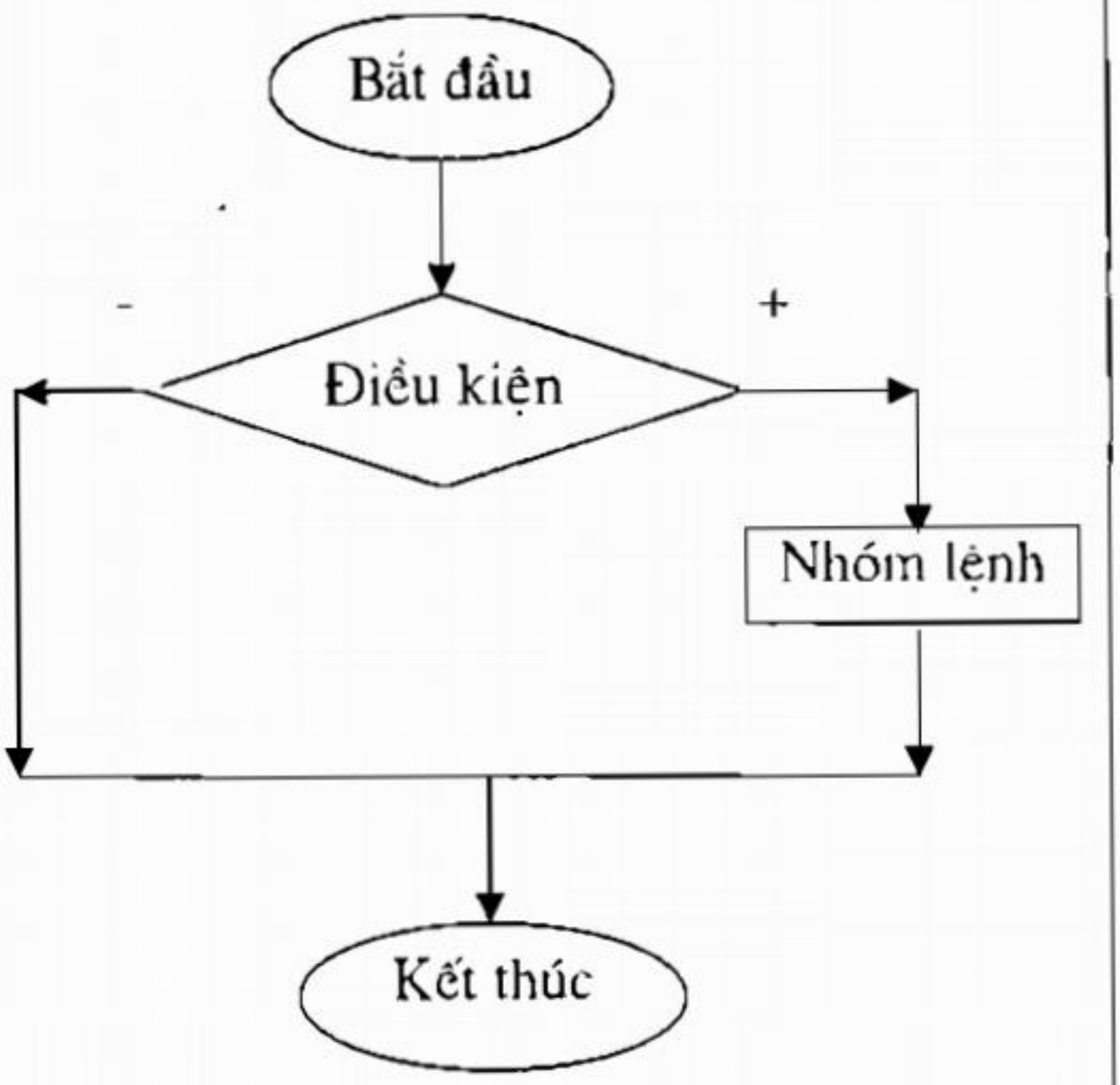
2.1. Cú pháp và hoạt động của lệnh rẽ nhánh IF

Cấu trúc rẽ nhánh trong ngôn ngữ Pascal nói riêng và các ngôn ngữ lập trình nói chung (giống như trong thuật toán) gồm hai loại: rẽ nhánh khuyết và rẽ nhánh đủ.

http://tieulun.hopto.org

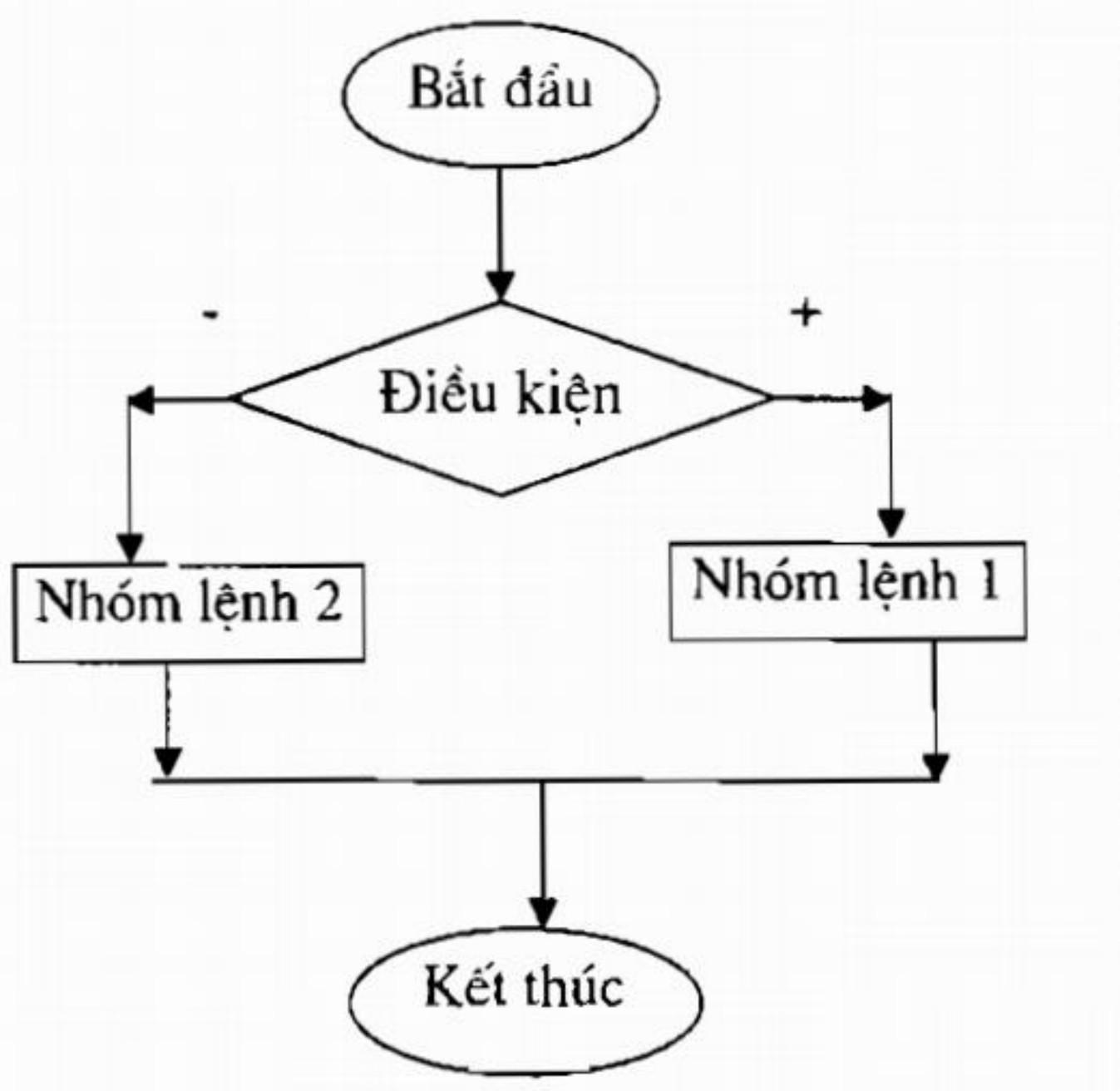
- *Rẽ nhánh khuyết:*

Cú pháp	Hoạt động của lệnh

IF điều_kiện Then
Begin
Nhóm_lệnh;
End;
	

- *Rẽ nhánh đủ*

Cú pháp	Hoạt động của lệnh

IF điều_kiện Then
Begin
Nhóm_lệnh1;
End
ELSE
Begin
Nhóm_lệnh2;
End;
	

http://tieulun.hopto.org

Chú ý: Nếu mỗi nhóm lệnh trong các cú pháp trên chỉ gồm một lệnh thì lệnh đó không cần viết trong câu lệnh ghép Begin ... End.

2.2. Câu lệnh ghép Begin... End

Để hiểu được ý nghĩa của câu lệnh ghép Begin...End ta hãy so sánh hai ví dụ sau đây:

Ví dụ 1	Ví dụ 2

If đk then
Begin
A;
B;
End;
C;
	If đk then
	A;
	B;
	C;

+ Nếu đk đúng thì A, B, C thực hiện. + Nếu đk sai thì C thực hiện. Vậy: A,B phụ thuộc vào đk, C không phụ thuộc vào điều kiện.	+ Nếu đk đúng thì A, B, C thực hiện. + Nếu đk sai thì B, C thực hiện. Vậy: A phụ thuộc vào đk, B và C không phụ thuộc vào điều kiện.

Ý nghĩa của khối lệnh begin.. end:

Trong một lệnh có cấu trúc (ví dụ như lệnh IF), nếu có từ hai lệnh trở lên cùng phụ thuộc vào một điều kiện nào đó (nghĩa là các lệnh này được thực hiện hay không được thực hiện tuỳ thuộc vào giá trị đúng hay sai của điều kiện) thì các lệnh này phải được viết trong câu lệnh ghép Begin... End;

Về mặt hình thức, các lệnh được nhóm lại bởi Begin... End thì nên viết lùi vào phía trong để chứng tỏ nó không bình đẳng với các lệnh nằm ngoài Begin... End;

Chú ý:

+ Lệnh đơn giản (nhắc lại) là một lệnh gán, lệnh vào/ra, lệnh gọi thực hiện một hàm hay thủ tục.

+ Nhóm lệnh trong cú pháp của lệnh If mà ta xét ở trên có thể bao gồm: các lệnh đơn giản, các lệnh có cấu trúc. Lệnh If là một lệnh có cấu trúc. Khi có một lệnh "If con" trong nhóm lệnh, người ta hay nói rằng có một cấu trúc rẽ nhánh lồng nhau.

http://tieulun.hopto.org

+ Câu lệnh If dù là dạng khuyết hay dạng đủ luôn luôn chỉ là một lệnh, mặc dù nhóm lệnh sau IF và sau Else (nếu có Else) có thể là khá nhiều lệnh và các lệnh đó là lệnh đơn giản hay lệnh có cấu trúc.

Trong các ví dụ sau đây, các chữ cái in hoa ký hiệu cho một lệnh đơn giản hay một lệnh có cấu trúc nào đó.

Ví dụ 3:

```
If đk then
Begin
  A; B;
End
Else
  C; D; E;
```

Ví dụ 3 có ba lệnh: Lệnh thứ nhất là lệnh If dạng đủ, hai lệnh còn lại là lệnh D và E.

Ví dụ 4:

```
If đk1 then A
Else
  Begin
    B;
    C;
    D;
  End
Else
  Begin
    M;
    N;
  End;
P;
```

Ví dụ 4 gồm hai lệnh: lệnh thứ nhất là lệnh If (dạng đủ) và lệnh thứ hai là lệnh P.

2.3. Ví dụ bài tập về lệnh rẽ nhánh if

Ví dụ 2.3

Viết chương trình giải bất phương trình bậc nhất:

http://tieulun.hopto.org

ax + b < 0 biết giá trị a, b nhập từ bàn phím.

GPTB1.PAS

```
Var  a,b: real;
Begin
   write('nhap a,b: '); Readln(a,b);
   if a = 0 then
     if b <= 0 then
          Writeln('BPT VD')
     else Writeln('BPT VN')
   else
   if a>0 then
       Writeln('x < ', -b/a:0:3)
   else Writeln('x > ', -b/a:0:3);
   Readln;
End.
```

Ví dụ 2.4

Viết chương trình nhập vào một tháng của một năm nào bất kỳ, sau đó máy tính in lên màn hình số ngày của tháng đó.

Năm nhuận là một năm chia hết cho 4, nhưng nếu đó là năm đầu thế kỷ thì phải chia hết cho 400. Ví dụ các năm 16, 100, 2004 là các năm nhuận, các năm 1900, 2001 không là năm nhuận.

Chương trình: NAMNHUAN.PAS

```
Var d,m,y: integer;
Begin
  Write('Nhap thang: '); Readln(m);
Write('Nhap nam: '); Readln(y);
d:=0;
if m in [1,3,5,7,8,10,12] then d:=31
else if m in [4,6,9,11] then d:=30
else if m =2 then
     begin
      if y mod 100 = 0 then
       if y mod 400 = 0 then d:=29
```


http://tieulun.hopto.org

```
                else d:=28
          else
           if y mod 4 = 0 then d:=29
                  else d:=28;
        end
    else write('Nhap sai thang') ;
    if d>0 then
writeln('Thang ',m,' nam ',y,' co so ngay la: ',d);
    readln;
End.
```

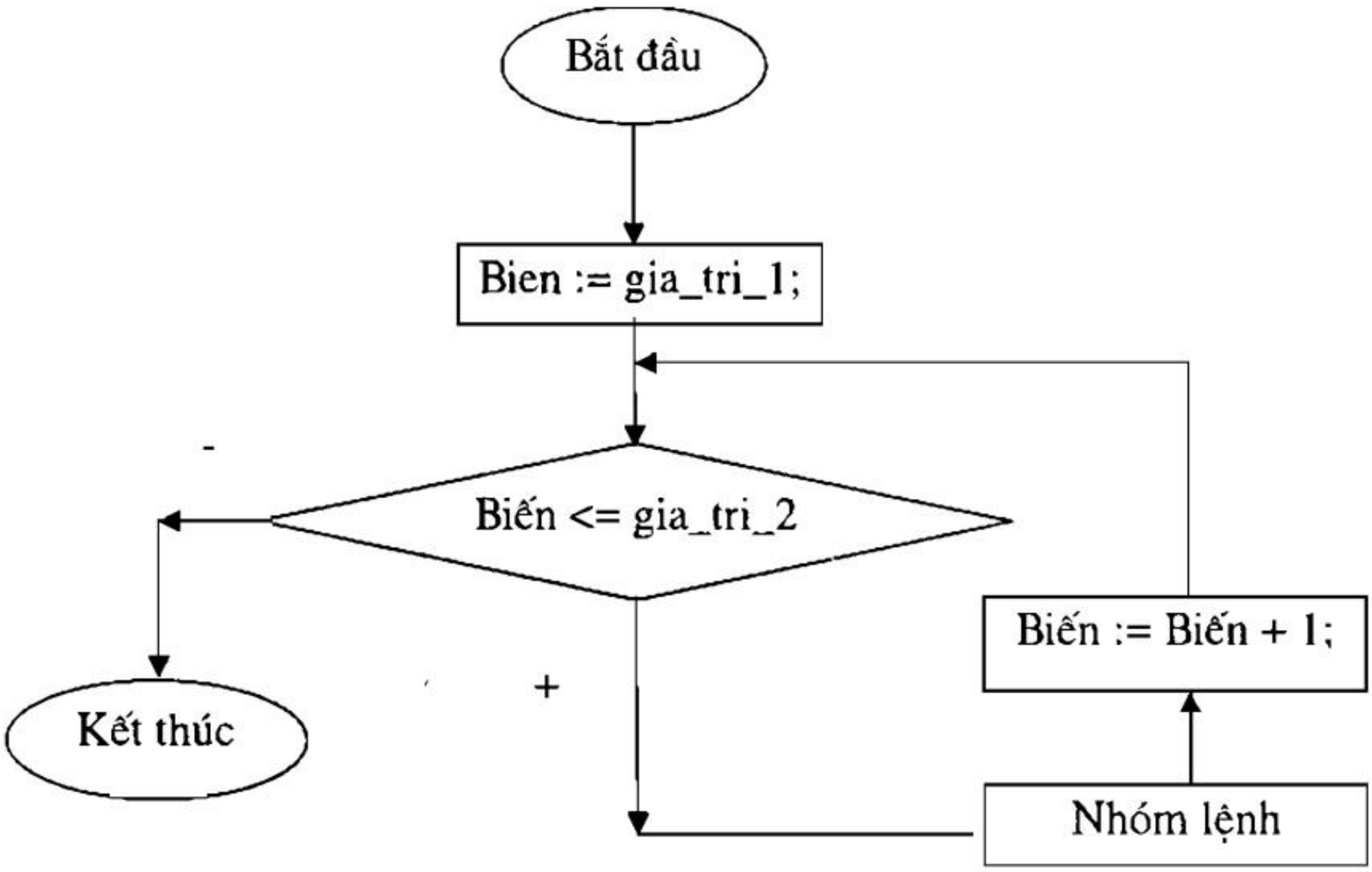
3. Cấu trúc lặp với số lần biết trước

3.1. Cú pháp và hoạt động của lệnh lặp for

Cú pháp:

```
FOR biến:= giá_tri_1 TO giá_tri_2 DO
Begin
   Nhóm_lệnh;
End;
```

Hoạt động của lệnh lặp for:



http://tieulun.hopto.org

Nhận xét: Biến có vai trò như một biến đếm, đếm số lần lặp của nhóm lệnh.

Xét số lần lặp:

giá trị 1	giá trị 2	số lần
1	10	10
0	10	11
0	9	10
2	10	9
-10	10	21
G1	G2	G2-G1+1

Vậy: Tổng số số lần lặp là 1 + |giá_trị_2 - giá_trị_1|

Khi giá_trị_1 >= giá_trị_2 ta thay TO bằng DOWNTO.

Khi giá_trị_1 = giá_trị_2 thì nhóm lệnh lặp thực hiện 1 lần.

3.2. Ví dụ bài tập về lệnh lặp for

Ví dụ 2.5

Viết chương trình tính và in lên màn hình giá trị biểu thức $T = a^n$ biết a, n là các số nguyên nhập từ bàn phím, n>0.

Phân tích thuật toán:

Ta có: $T = a^n$ hay

T = a*a*....*a, có n thừa số a trong tích.

Nếu ban đầu ta gán T := 1 thì tiếp theo lệnh T := T*a sẽ thực hiện n lần để có kết quả $T=a^n$.

Đoạn trình cốt lõi (Đoạn chương trình ứng với phần cốt lõi của thuật toán)

```
T := 1;
For i:=1 to n do T:= T * a;
Write('T = ',T);
```

Chương trình: LUYTHUA.PAS

```
Var   i,a,n: integer;
      T: longint;
Begin
 Write('a = '); Readln(a);
```


http://tieulun.hopto.org

```
  Write('n = '); Readln(n);
   T:= 1;
   For i:= 1 to n do T:= T * a;
   Writeln(a, '^', n, ' = ',T);
  Readln;
End;
```

Ví dụ 2.6

Viết chương trình tính và in lên màn hình giá trị biểu thức P = n! với giá trị của n nhập từ bàn phím, n ≥ 0.

Phân tích thuật toán:

Định nghĩa n!=1*2*...*n;

Ta có 0!=1 theo quy ước:

1!=1*1 = 0!*1

2!=1*2 = 1!*2

3!=(1*2)*3=2!*3

4!=(1*2*3)*4 = 3!*4

Vậy n!=(n-1)!*n

Từ đó nếu ban đầu đặt P := 1 thì tiếp theo lệnh P := P*i sẽ thực hiện n lần với i chạy từ 1.2,..., n để có kết quả P = n!.

Thật vậy, với i = 1 ta có P = 1 = 1!. Với i = 2, ta có P = 1!*2 = 2!. Với i = 3, ta có P = 2!*3 = 3!... Với i = n, ta có P = (n-1)!*n = n!.

Đoạn trình cốt lõi:

```
P := 1;
For i:=1 to n do P:= P * i;
Write('P = ',P);
```

Chương trình: GIAITHUA.PAS

```
Var  i,n: integer; P: Longint;
Begin
  Write('n = '); Readln(n);
  P:= 1;
  For i:= 1 to n do P:= P * i;
  Writeln(n,' != ', P); Readln;
End;
```


http://tieulun.hopto.org

Ví dụ 2.7

Tính và in lên màn hình giá trị xấp xỉ biểu thức pi/4 theo công thức:

$Q = 1 - 1/3 + 1/5 - 1/7 + ... + (-1)^n/(2n-1)$

với giá trị của số nguyên dương n nhập từ bàn phím.

Phân tích thuật toán:

Ban đầu Q = 0.

Giả sử d bằng +1 hay -1 để biểu diễn dấu của các số hạng trong tổng Q. Ban đầu, ứng với dấu dương của số hạng đầu tiên, ta khởi tạo d bằng +1.

Để tính tổng Q, xét các mẫu số của các số hạng trong Q, ta sẽ cho biến i "chạy" từ 1 đến n và với mỗi giá trị của i , ta thêm vào tổng Q số hạng:

d*1/(2*i-1)

đồng thời đổi dấu d :=-d để d trở thành dấu của số hạng tiếp theo.

Bảng mô phỏng với n = 4. Ta có Q = 1 - 1/3 + 1/5 - 1/7.

i	số hạng d * 1/(2*i-1)	Q := Q + d*(1/2*i-1)	dấu d cho lần sau
Khởi tạo		0	1
1	1/1	1	-1
2	-1/3	(1)-1/3	1
3	1/5	(1-1/3)+1/5	-1
4	-1/7	(1-1/3+1/5)-1/7	

Đoạn trình cốt lõi:

```
Q := 0; d := 1;
For i:=1 to n do
begin
   Q := Q * d*(1/2*i - 1);
   d := -d ;
end;
```

Chương trình: TINHTONG.PAS

```
Var  n,i,d: integer;
       Q: Real;
Begin
   Write('n = '); Readln(n);
```


http://tieulun.hopto.org

```
{giả thiết nhập đúng đắn n là số dương}
 Q:= 0;  d:=1;
 For i:= 1 to n do
 Begin
      Q:= Q + d*1/(2*i-1);
      d:= - d;
 End;
Writeln(' Q= ', Q:10:3);
 Readln;
End.
```

Ví dụ 2.8

Viết chương trình tìm số hạng thứ n của dãy Fibonaci.

Dãy fibonaci là dãy mà số hạng thứ nhất và thứ hai bằng 1, bắt đầu từ số hạng thứ 3 trở đi bằng tổng của hai số hạng đứng ngay trước nó.

Bảng sau đây ví dụ bẩy số hạng Fibonaci đầu tiên, trong đó f(i) là số hạng thứ i của dãy Fibonaci.

i	1	2	3	4	5	6	7
f(i)	1	1	2	3	5	8	13

Phân tích thuật toán:

Gọi f1,f2,f3 là giá trị của 3 số hạng fibonaci liên tiếp. Trong đó, f3 là số hạng fibonaci thứ i, còn f2,f1 là hai số hạng trước đó.

Trường hợp riêng với n = 1 hoặc 2 thì f3 = 1.

Trường hợp tổng quát với n >= 3. Ta sẽ cho i chạy từ 3 đến n, với mỗi giá trị của i ta sẽ xác định f3 - số hạng Fibonaci thứ i. Khi i đạt đến n thì f3 là số hạng Fibonaci thứ n cần tìm. Cụ thể như sau:

Ban đầu khởi tạo f1= f2 =1.

Với i = 3 thì f3 = f1+f2=2. Để tính tiếp số hạng thứ 4, ta "tịnh tiến" f1,f2 sang phải một vị trí, tức là f1:= f2; f2 := f3. Vậy lúc này f1 = 1, f2 = 2.

Với i = 4 thì f3 = f1+f2= 3. Để tính tiếp số hạng thứ 5, ta lại tịnh tiến f1,f2 sang phải một vị trí, f1:= f2; f2 := f3. Vậy lúc này f1=2, f2=3.

Với i=5, thì f3=f1+f2=5....

http://tieulun.hopto.org

Cứ tiếp tục cách như vậy đến khi i bằng n thì f3 là số hạng thứ n cần tìm.

Một cách tổng quát, với một giá trị nào đó của i>= 3 thì f3 = f1+f2 là giá trị của số hạng thứ i. Để chuẩn bị tính số hạng thứ i+1, ta dịch chuyển f1,f2 sang phải một vị trí f1:=f2; f2:=f3. Bây giờ chính thức xét số hạng thứ i+1 giống như cách làm đối với số hạng thứ i.

Bảng mô phỏng với n = 7:

i	f3	f1	f2
Ban đầu		1	1
3	2	1	2
4	3	2	3
5	5	3	5
6	8	5	8
7	13	8	13

Chương trình: FIBONACI.PAS

```
Var f1,f2,f3: integer;
    i,n: i nteger;
Begin
Write (' Tim so hang thu bao nhieu: ') ; Readln(n);
If (n=1) or (n=2) Then f3:=1
Else
Begin
  f1:=1; f2:=1;
  For i:= 3 to n do
  Begin
       f3:= f1+f2;
       f1:= f2; f2:= f3;
    End;
End;
Writeln('So hang thu ',n, ' = ',f3);Readln;
End.
```


http://tieulun.hopto.org

Ví dụ 2.9

Viết chương trình kiểm tra một số a nguyên dương cho trước có phải là số nguyên tố hay không.

Phân tích thuật toán:

a là một số nguyên tố nếu nó không có ước thực sự (không có ước nào khác ngoài hai ước tầm thường là 1 và chính nó).

Nói cách khác a là số nguyên tố nếu số lượng các ước (không kể chính nó) đúng bằng 1 (đó chính là ước bằng 1 duy nhất). Ta sẽ sử dụng tính chất này để kiểm tra xem số a có là số nguyên tố hay không.

Gọi d là số lượng các ước của a (không kể a), ban đầu khởi tạo d = 0;

Để tính d, ta sẽ dùng phép thử. Nghĩa là xét với mọi i **có khả năng** là ước của a. Như vậy i nhỏ nhất là 1 và cùng lắm i = a-1, đánh giá chặt hơn nữa, i chỉ cần "chạy" đến a div 2 (nghĩa là đến giá trị nguyên lớn nhất của i mà tại đó i*i ≤ a). Trong quá trình i chạy từ 1 đến a div 2, nếu giá trị nào của i là ước của a thì ta đếm bằng cách tăng 1 đơn vị cho d.

Kết thúc quá trình duyệt, nếu d = 1 thì chứng tỏ có duy nhất một giá trị của i =1 là ước của a, nói cách khác, a là số nguyên tố. Nếu ngược lại, (d>1) thì a là hợp số.

Lý do mà i phải chạy từ 1 là vì ta muốn xét tổng quát gồm các số nguyên tố từ 2 trở đi. Nếu cho i chạy từ 2 thì các số nguyên tố cần kiểm tra phải > 2 và khi đó nếu d = 0 thì số cần kiểm tra là số nguyên tố.

Chương trình: PRIME.PAS

```
Var a,i,d: integer;
Begin
  Write('Nhap so duong a: '); Readln(a);
  d:= 0 ;
  for i:=1 to a div 2 do
  if a mod i = 0 then d:= d + 1;
  if d = 1 then
       Writeln(a,' la so nguyen to ')
  else Writeln(a,' la hop so ');
Readln;
End.
```


http://tieulun.hopto.org

Ví dụ 2.10

Viết chương trình kiểm tra một số a nguyên dương cho trước có phải là số hoàn thiện hay không. Ta nói rằng a là số hoàn thiện nếu nó bằng tổng các ước của nó.

Phân tích thuật toán:

Ta hoàn toàn có thể liên hệ (tương tự) thuật toán kiểm tra tính nguyên tố của số nguyên a sang thuật toán kiểm tra tính hoàn thiện của a. Cụ thể, thuật toán kiểm tra tính hoàn thiện của số nguyên a có được bằng cách sửa lại thuật toán kiểm tra tính nguyên tố của số nguyên a như sau:

Thay đổi vai trò của d lúc này không dùng để đếm số lượng các ước (< a) của a mà được sử dụng để tính tổng các ước (< a) của a.

Vậy có 2 chỗ cần sửa lại:

Thay lệnh đếm d := d+1 bằng lệnh tích lũy ước d:= d+i ;

Thay điều kiện d = 1 bằng điều kiện d = a.

Chương trình:

PERFECT.PAS

```
Var a,i,d: integer;
Begin
  Write('Nhap so duong a: '); Readln(a);
  d:= 0 ;
  for i:=1 to a div 2 do
  if a mod i = 0 then d:= d + i;
  if d = a then
       Writeln(a,' la so hoan thien ')
  else Writeln(a,' khong la so hoan thien');
Readln;
End.
```

3.3. Lệnh ngắt Break

Lệnh ngắt break có thể được dùng trong các câu lệnh lặp của ngôn ngữ Pascal để làm cho vòng lặp sớm kết thúc. Lệnh break rất hay được sử dụng trong câu lệnh lặp for.

http://tieulun.hopto.org

Trong nhóm lệnh lặp của for, nếu ta đưa vào một lệnh break thì khi thực hiện lệnh break, lệnh for sẽ kết thúc ngay và số lần lặp không nhất thiết như đã tính ở trên. Chẳng hạn, ta cần dùng lệnh lặp for để duyệt trên một dãy các phần tử để kiểm tra xem có hay không có một phần tử trong dãy thoả mãn một tính chất nào đó. Nếu trong khi duyệt, gặp được phần tử này, ta sẽ cho thực hiện lệnh break để vòng lặp for sớm kết thúc. Như vậy thời gian thực hiện chương trình sẽ giảm đi đáng kể.

Ví dụ 2.11

Chương trình sau đây mô phỏng việc máy tính gieo xúc xắc một cách ngẫu nhiên tối đa n lần. Trong quá trình gieo xúc xắc, nếu gặp được mặt 6 chấm thì dừng lại không gieo tiếp nữa.

Chương trình: XUC_XAC.PAS

```
Uses crt;
Var  i,n, a : integer;
Begin
 Write('So lan gieo xuc xac toi da: '); Readln(n);
 randomize; { khởi tạo bộ sinh số ngẫu nhiên}
 For i:= 1 to n do
 begin
  a:= random(6) + 1; {sinh một số ngẫu nhiên trong đoạn 1..6 }
   if a = 6 then break;
 end;
 if i < = n then Writeln('Gap duoc mat 6')
 Else writeln('Khong gap duoc mat 6');
 Readln;
End;
```

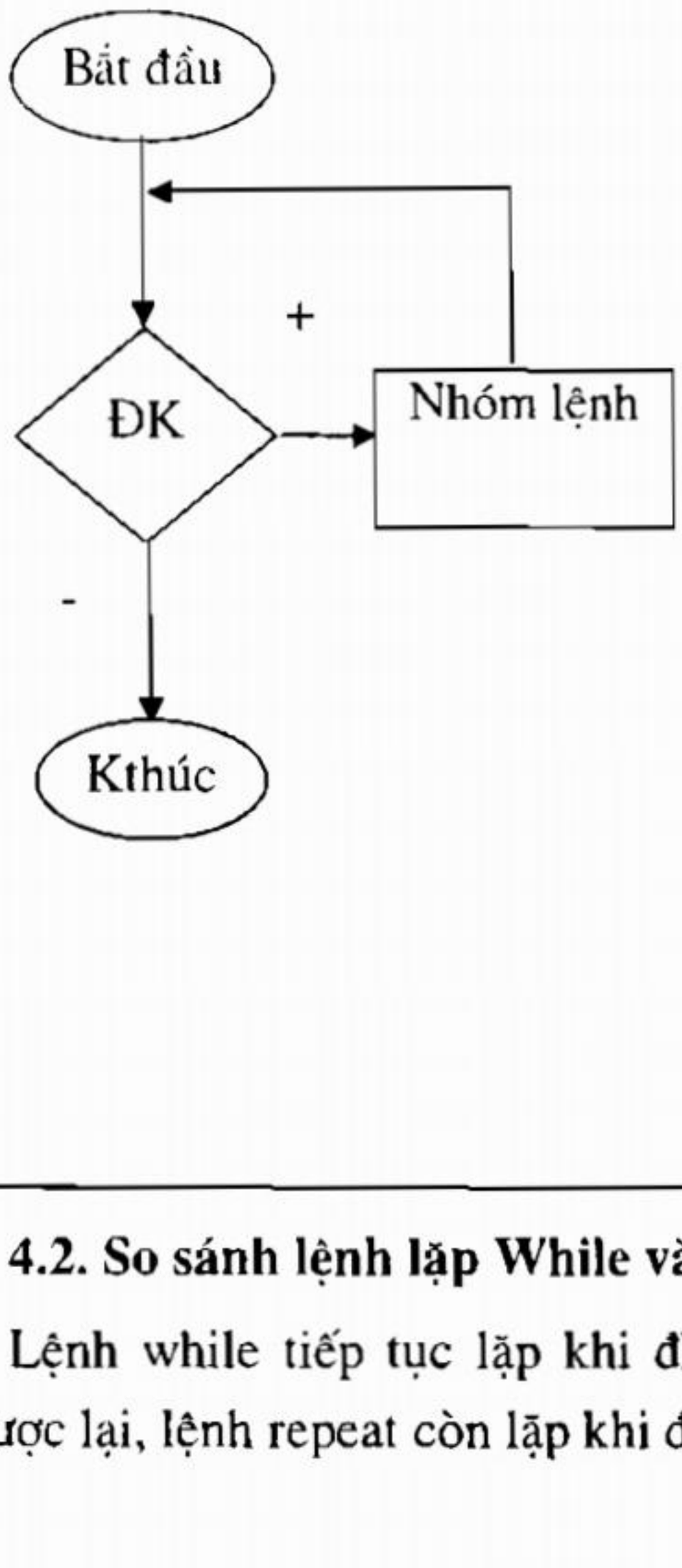
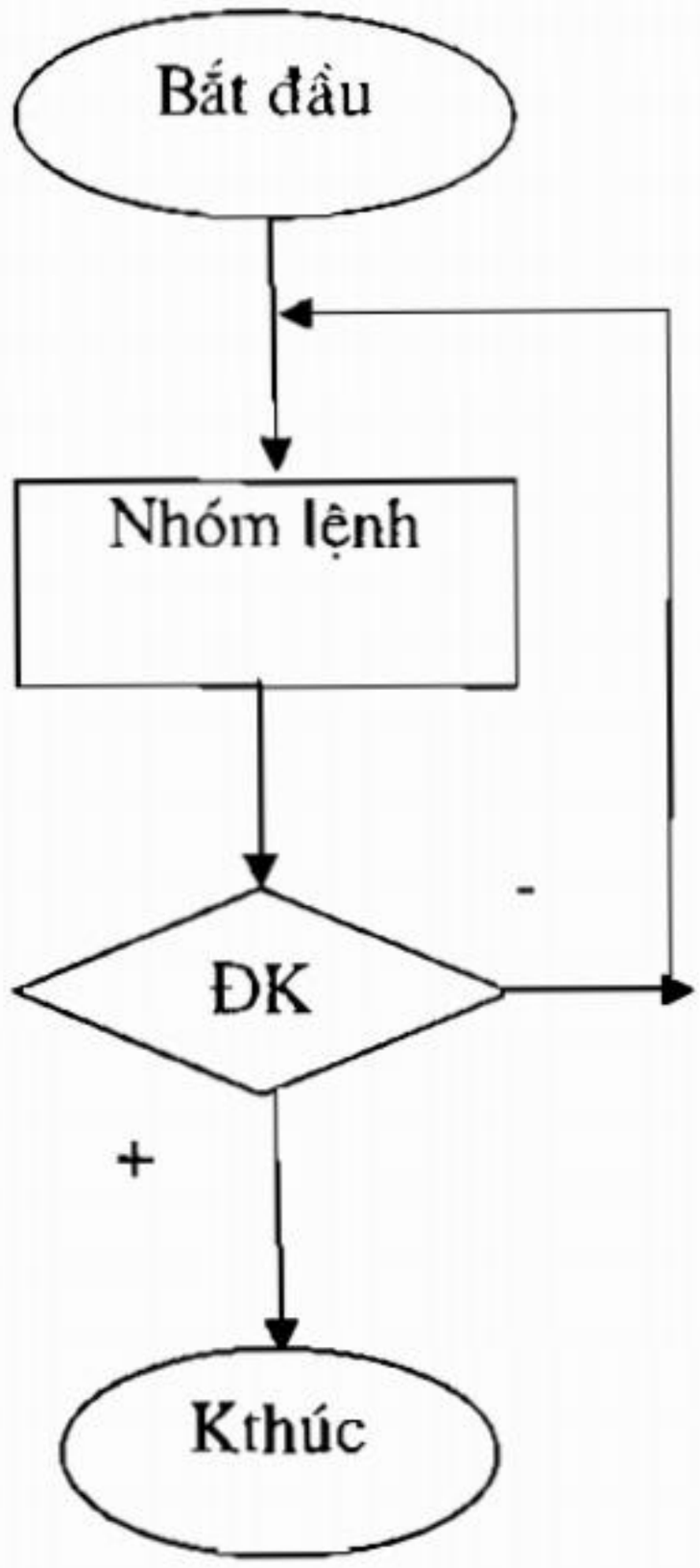
4. Cấu trúc lặp với số lần không biết trước

4.1. Cú pháp và hoạt động của hai lệnh lặp While và Repeat

Ngôn ngữ Pascal có hai cấu trúc lặp và không biết trước số lần lặp là lệnh lặp While và lệnh lặp Repeat. Có một điều kiện (biểu thức logic) để điều khiển vòng lặp, nghĩa là tuỳ theo giá trị đúng hay sai của điều kiện mà

http://tieulun.hopto.org

quá trình lặp sẽ kết thúc hoặc tiếp tục thực hiện lại nhóm lệnh nào đó lần nữa. Như vậy trước khi bắt đầu thực hiện câu lệnh lặp, điều kiện này phải có giá trị xác định và quan trọng hơn nữa là: trong nhóm lệnh lặp phải có một lệnh làm thay đổi giá trị của điều kiện đó để đảm bảo không xảy ra quá trình lặp vô hạn lần.

<table>
<tr><th>Lặp kiểm tra điều kiện trước</th><th>Lặp kiểm tra điều kiện sau</th></tr>
<tr><td>Whilc đk do
Begin
Nhóm_lệnh;
End;</td><td>Repeat
Nhóm_lệnh;
Until đk;</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

4.2. So sánh lệnh lặp While và lệnh lặp Repeat

Lệnh while tiếp tục lặp khi đk (điều kiện) đúng (dừng lặp khi đk sai). Ngược lại, lệnh repeat còn lặp khi đk sai (dừng lặp khi điều kiện đúng).

http://tieulun.hopto.org

Vì kiểm tra đk trước nên nhóm lệnh lặp trong câu lệnh lặp while có thể không thực hiện lần nào, đó là trường hợp mà ngay từ trước khi thực hiện lệnh while thì đk đã có giá trị đúng. Ngược lại, vì kiểm tra đk sau nên câu lệnh lặp repeat sẽ cho phép nhóm lệnh thực hiện ít nhất một lần, cho dù ngay từ đầu đk đó đã có giá trị đúng.

4.3. Ví dụ bài tập về hai lệnh lặp While và Repeat

Ví dụ 2.12

Viết chương trình tìm và in lên màn hình USCLN của hai số nguyên a, b nhập từ bàn phím.

Thuật toán tìm USCLN lớn nhất của hai số nguyên a và b đã được trình bầy trong ví dụ 1.10. Sau đây là chương trình cài đặt thuật toán đó.

Phương pháp trừ liên tiếp

```
Var a,b: integer;
    a1,b1: integer;
Begin
   Write('Nhap so a: '); Readln(a);
   Write('Nhap so b: '); Readln(b);
   a1:= a; b1:= b;
   a:= abs(a); b:= abs(b);
   while a<> b do
   if a>b then a:=a-b
   else b:=b-a;
   Writeln('(' ,a1,',' ,b1, ') = ', b);
   Readln;
End.
```

Phương pháp chia liên tiếp

```
Var a,b: integer;
    a1,b1,r: integer;
Begin
   Write('Nhap a va b: '); Readln(a,b);
   a1:= a; b1:= b; a:= abs(a); b:= abs(b);
   r:= a mod b;
   while r<>0 do
   begin
```


http://tieulun.hopto.org

```
        a:=b;
        b:=r;
        r:= a mod b;
    end;
    Writeln('(' ,a1, ',' ,b1, ') = ', b);
    Readln;
End.
```

Ví dụ 2.13

Viết chương trình giải bài toán sau: Một người gửi tiết kiệm số tiền ban đầu là a (triệu đồng) lãi suất k % /1 tháng, hãy cho biết người đó phải gửi bao nhiêu tháng để thu được số tiền là b triệu đồng. Các số thực dương a, k và b nhập từ bàn phím.

- *Phân tích thuật toán*

Giả sử S là số tiền cả gốc lẫn lãi sau khi gửi tiết kiệm t tháng.

Ban đầu chưa gửi tháng nào thì t = 0 và S = a.

Cứ sau mỗi tháng thì ta tăng biến đếm t lên một đơn vị, đồng thời thêm số tiền lãi S*k / 100 vào cho tổng S.

Vậy khi số tiền S còn chưa đạt đến b thì còn lặp lại hai lệnh sau đây:

t := t + 1 ; S := S + S *k /100;

- *Đoạn trình cốt lõi*

```
t:=0;  s:=a;
    while s<b do
    begin
        t:=t+1;
        s:= s + s * k/100;
    end;
```

Chương trình: TIETKIEM.PAS

```
Var  a,k,b: real;
     t: longint;
     s: real;
Begin
   Repeat
     Write('Nhap so tien ban dau a: '); Readln(a);
```


http://tieulun.hopto.org

```
Until (a>0);
Repeat
  Write('Nhap lai suat k: '); Readln(k);
Until (k>0);
Repeat
  Write('Nhap so tien can rut b: '); Readln(b);
Until (b>=a);
t:=0; s:=a;
while s<b do
begin
   t:=t+1;
   s:= s + s * k/100;
end;
Writeln(' so thang can gui la ', t);
Readln;
End.
```

5. Câu hỏi và bài tập

5.1. Bài tập về cấu trúc rẽ nhánh If

Bài 2.10

Viết chương trình giải phương trình $ax^2 + bx + c = 0$ với các số thực a,b,c nhập từ bàn phím.

Bài 2.11

Viết chương trình giải hệ phương trình:

$$\begin{cases} a1x + b1y = c1 \\ a2x + b2y = c2 \end{cases}$$

với các số thực a1, b1, c1, a2, b2, c2 nhập từ bàn phím.

Bài 2.12

Viết chương trình tính tiền điện tiêu thụ của một hộ gia đình biết chỉ số điện kế của tháng trước và tháng hiện tại và đơn giá điện trên 1 KWh quy định như sau: Đơn giá =

- 400 đ/KWh: điện tiêu thụ không quá 100 KW.
- 500 đ/KWh: điện tiêu thụ trên 100 đến không quá150 KW.
- 800 đ/KWh: điện tiêu thụ trên 150 đến không quá 200 KW.
- 1000 đ/KWh: điện tiêu thụ trên 200 KW.

http://tieulun.hopto.org

5.2. Bài tập về cấu trúc lặp For

Bài 2.13

In lên màn hình tất cả các số nguyên tố trong đoạn từ 1 đến 10.000.

Bài 2.14

In lên màn hình các cặp số hoàn thiện trong đoạn [a,b]. Các số nguyên a,b nhập từ bàn phím.

Bài 2.15

Một người gửi tiết kiệm với số tiền ban đầu là a (triệu đồng) và lãi suất hàng tháng là k%. Hãy cho biết số tiền thu được cả gốc lẫn lãi của người đó sau khi gửi t tháng? Các số thực dương a, k, t nhập từ bàn phím.

Bài 2.16

Viết chương trình giải bài toán:

"Vừa gà, vừa chó, bó lại cho tròn, 36 con, 100 chân chẵn. Hỏi có bao nhiêu con gà, bao nhiêu con chó "?

Bài 2.17

Viết chương trình giải bài toán:

"Trăm trâu trăm cỏ, trâu đứng ăn 5, trâu nằm ăn 3, lụ khụ trâu già 3 con một bó. Hỏi trâu đứng, trâu nằm, trâu già mỗi loại có bao nhiêu con"?

Bài 2.18

In lên màn hình tất cả các số có 4 chữ số mà các chữ số của nó có tổng và tích bằng nhau.

Bài 2.19

Lập trình mô tả trò chơi thử vận may theo mô tả như sau: Đầu tiên máy phát sinh một số ngẫu nhiên n để biểu thị có n lượt chơi. Ở mỗi lượt chơi, máy sẽ phát sinh một số ngẫu nhiên, tuy nhiên trước khi thông báo số này, người chơi sẽ phải đoán trước xem số này chẵn (nhập 0) hay số lẻ (nhập 1). Kết thúc n lượt đi, máy tính sẽ thông báo tổng số lần đoán đúng và tỷ lệ đoán đúng, từ đó máy tính thông báo về khả năng may mắn của người chơi theo quy định như sau:

Nếu tỷ lệ đoán đúng là:

- Từ 80% trở lên thì "rất may mắn".
- Từ 65% đến dưới 80% thì "may mắn".

http://tieulun.hopto.org

- Từ 50% đến dưới 65% thì "bình thường".
- Dưới 50% thì "không may mắn".

5.3. Bài tập về cấu trúc lặp While và Repeat

Bài 2.20

Dân số nước ta năm 1999 là a triệu người, tốc độ tăng dân số là k% 1 năm. Hãy cho biết đến năm bao nhiêu thì dân số nước ta tăng gấp p lần so với năm 1999? Các số a, k và p nhập từ bàn phím.

Bài 2.21

Tìm số n nhỏ nhất để giá trị của biểu thức sau đây không nhỏ hơn 2 triệu.

$M = 1^2 + 2^2 + ... + n^2$

Bài 2.22 *

Trò chơi bốc sỏi giữa 2 đối thủ mô tả như sau: Ban đầu có n viên sỏi. Hai người chơi luân phiên, ai đến lượt mình đi sẽ bốc ít nhất là một viên sỏi nhưng nhiều nhất là k viên. Số k và n cho trước, nhập từ bàn phím.

Hãy viết chương trình mô tả trò chơi bốc sỏi giữa người và máy sao cho máy giành được nhiều phần thắng nhất? Biết rằng ai phải bốc viên sỏi cuối cùng là người thua cuộc.

http://tieulun.hopto.org

Chương 3

CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC CƠ BẢN

I. CẤU TRÚC DỮ LIỆU KIỂU MẢNG MỘT CHIỀU

1. Định nghĩa, khai báo mảng một chiều

1.1. Định nghĩa mảng một chiều

Mảng một chiều là một danh sách có thứ tự và hữu hạn các phần tử có cùng một kiểu dữ liệu.

1.2. Cách khai báo mảng một chiều

1.2.1. Khai báo trực tiếp

```
VAR
TênBiếnMảng: ARRAY[ kiểu_chỉ_số] OF Kiểu_phần_tử;
```

1.2.2. Khai báo gián tiếp

```
TYPE
TênKiểuMảng= ARRAY [ kiểu_chỉ_số] OF Kiểu_phần_tử;
VAR
Tên_biến_mảng: tên_kiểu_mảng;
```

Kiểu phần tử của mảng là kiểu dữ liệu quy định cho từng phần tử trong mảng. Kiểu chỉ số của mảng được mô tả bởi một kiểu đoạn con và xác định vị trí của các phần tử trong mảng. Đồng thời, kiểu chỉ số của mảng còn quy định số lượng tối đa các phần tử trong mảng.

Kiểu đoạn con để mô tả kiểu phần tử của mảng được chỉ ra bằng một giá trị làm cận dưới, tiếp theo là hai dấu chấm (có đúng hai dấu chấm) và cuối cùng là giá trị cận trên. Có thể khai báo trước một kiểu đoạn con, sau đó mới khai báo mảng như sau:

http://tieulun.hopto.org

```
TYPE
TênKiểuĐoanCon=GiáTrị_CậnDưới .. GiáTrị_CânTrên;
VAR
Tên_biến_mảng: ARRAY [ TênKiểuĐoanCon] OF Kiểu_phần_tử;
```

Tổng dung lượng bộ nhớ dành cho các biến toàn cục trong đó có mảng là không vượt quá 64KB (làm tròn của 65536 bytes). Như vậy, giả sử biến toàn cục chỉ có một mảng với kiểu phần tử là Byte thì đoạn con dài không vượt quá 64000.

Khi khai báo mảng, thường có thêm 2 biến:

n: để chỉ rõ số phần tử thực sự được xét trong mảng.

i : để xác định vị trí của các phần tử trong mảng.

1.2.3. Ví dụ về khai báo mảng một chiều

```
TYPE
  INDEX = 1..60;
  DIEM = Array [ index] of real;
VAR
  KV1, KV2: DIEM;
  Code: Array [ Index] of Char;
  A,B : Array [ -100..100] of integer;
  P,Q: Array [ 'A' ..' Z'] of Byte;
```

2. Vào/ra mảng một chiều

2.1. Phép truy nhập mảng

Không thể vào/ra cả một biến mảng bằng lệnh readln và writeln.

Xét các mảng trong ví dụ trên thì các lệnh sau đây là sai:

```
Readln(A); Writeln(A);
```

Chỉ có thể truy nhập từng phần tử của mảng thông qua tên biến mảng và vị trí của nó đặt trong dấu [].

Tuy nhiên hai biến mảng cùng kiểu thì có thể gán trực tiếp cho nhau.

Xét tính đúng sai của các lệnh sau đây:

Lệnh	Nhận xét
KV1 [3]:= 9.5	đúng
KV1 [65]:= 7.5	sai về chỉ số, vượt quá chỉ số

http://tieulun.hopto.org

code[59]:= 'Nga'	sai về kiểu phần tử, phải là kiểu char
A[0]:= 5.3	sai về kiểu phần tử, phải là kiểu integer
B[-3]:= 6;	đúng
P[2]:= 260;	sai về kiểu chỉ số và kiểu phần tử
P[B]:= 200;	sai về kiểu phần tử nếu B là biến khác kiểu char
P['B']:= 200;	đúng

2.2. Mẫu nhập mảng một chiều

Để cho dễ hiểu, việc xem xét các đoạn trình mẫu và ví dụ về duyệt trên mảng một chiều (nhập mảng, in mảng và duyệt mảng) ta sử dụng chung khai báo sau đây:

```
CONST maxN = 100;
VAR A   : Array[1..maxN] of real;
    n,i : integer;
```

Trong khai báo trên ta có thể giả sử mảng A biểu diễn điểm của n học sinh trong một lớp (sĩ số của lớp không quá 100 em). Điểm được quy định theo thang điểm 10 và chính xác đến 0.25. Các học sinh có số hiệu là 1,2,..., n.

2.2.1. Nhập mảng một chiều có điều kiện

Đoạn chương trình sau đây làm nhiệm vụ nhập sĩ số lớp và điểm của n học sinh trong lớp đó.

```
Repeat
  Write('n = ');
Readln(n);
Until (n>=1) and (n<=maxN);
For i:=1 to n do
Repeat
  write('a[' ,i,  ']= ');
  Readln(a[i]);
Until (a[i]>=0) and (a[i]<=10)and(round(a[i]*100)mod 25= 0);
```

2.2.2.Nhập mảng một chiều không có điều kiện

Đoạn trình trên đảm bảo người chạy chương trình phải nhập A[i] là một số thực dương biểu thị phần thập phân chính xác đến 0.25 (i=1,2,...,n).

http://tieulun.hopto.org

Nếu mảng A chỉ biểu diễn dãy số đơn thuần, không có quy định điều kiện cụ thể đối với A[i] thì trong đoạn trình trên ta sẽ thay lệnh Repeat .. Until bằng câu lệnh ghép Begin ... End;

```
Repeat
  Write('n = ');
Readln(n);
Until (n>=1) and (n<=maxN);
For i:=1 to n do
Begin
  write('a[' ,i,  ']= ');
  Readln(a[i]);
End;
```

2.3. Mẫu in mảng lên màn hình

For i:=1 to n do Write(A[i],' '); Writeln;

3. Duyệt mảng và kỹ thuật 3 bước trong lập trình

3.1. Duyệt mảng tổng quát

3.1.1. Duyệt một lần

```
For i:=1 to n do <xử lý A[i]>;
```

Hoặc, chi tiết hơn:

```
For i:=1 to n do
If < A[i] thỏa mãn điều kiện P > Then < xử lý A[i]>;
```

3.1.2. Duyệt so le (phương pháp đi 'cà kheo') để đối chiếu

```
For i:=1 to n-1 do
For j:=i+1 to n do
If <so sánh A[i] và A[j]> then <xử lý kéo theo>.
```

Mẫu (3.1.2) có nhiều biến dạng và sẽ được nghiên cứu sau.

Nhập mảng và in mảng chỉ là 2 trường hợp riêng của duyệt màng tổng quát.

3.2. Một số đoạn trình ví dụ về duyệt mảng

Ví dụ 1:

Viết đoạn chương trình tính điểm trung bình của các học sinh trong lớp:

http://tieulun.hopto.org

```
S:=0;
For i:=1 to n do S:= S + A[i];
Writeln('Diem trung binh la: ', S / n: 5:2);
```

Ví dụ 2:

Viết đoạn chương trình tính tỷ lệ học sinh đỗ trong lớp:

```
d:=0;
For i:=1 to n do
if A[i]>=5.00 then d:= d + 1;
Writeln('Ty le do la: ' , d*100/n: 5:2 ,%');
```

Ví dụ 3:

Viết đoạn chương trình tính tỷ lệ học sinh giỏi, khá, trung bình, yếu kém trong lớp:

```
g:=0; k:=0; tb:=0; yk:= 0;
for i:=1 to n do
If A [i]>=8.0 then inc(g)
Else If A [i]>=6.5 then inc(k)
Else If A [i]>=5.0 then inc(tb);
yk:= n-g-k-tb;
Writeln( g*100/n:0:2,   ' % gioi ');
Writeln( k*100/n:0:2,   ' % kha ');
Writeln( tb*100/n:0:2,   ' % trung binh ');
Writeln( yk*100/n:0:2,   ' % yeu kem ');
```

3.3. Kỹ thuật 3 bước trong lập trình

Các ví dụ 1,2,3 vừa trình bày là các trường hợp đơn giản của **kỹ thuật 3 bước** trong lập trình. Kỹ thuật 3 bước được hình thức hóa như sau:

Bước 1: Khởi tạo;

Bước 2: Duyệt trên toàn bộ các thành phần của đối tượng đang xét. Trong quá trình duyệt có thể phải kiểm tra, đặt lọc để tính toán hoặc tích lũy đại lượng cần phải tính.

Bước 3: Nghiệm thu kết quả sau khi duyệt.

Một điều khá thú vị là "Kỹ thuật 3 bước" là một kỹ thuật đơn giản, đến mức dễ làm ta hiểu lầm rằng: khái quát lên như vậy chỉ làm "phức tạp hoá vấn

http://tieulun.hopto.org

đề". Thực ra, đây là một kỹ thuật rất quan trọng, hầu như các phép duyệt mảng nói riêng và các quá trình xem xét một cách toàn diện các đối tượng của bài toán đều sử dụng kỹ thuật này.

Các bài tập, ví dụ và các thuật toán tiếp theo về mảng một chiều, tất cả đều áp dụng kỹ thuật 3 bước ở những góc độ khác nhau.

4. Một số thuật toán tìm kiếm đơn giản

Tìm kiếm một hay nhiều phần tử trong mảng thoả mãn một tính chất nào đó là một vấn đề quan trọng trong lập trình vì nó có ý nghĩa rất thực tế. Ví dụ như tìm kiếm để lập ra một danh sách các thí sinh trúng tuyển (điểm trung bình ba môn thi ≥ 6) trong hàng nghìn thí sinh dự thi; tìm kiếm để lập danh sách các cán bộ đã đủ tiêu chuẩn về hưu (nam ≥ 60 tuổi, nữ ≥ 55 tuổi) trong một cơ quan nào đó,... Mỗi một đối tượng trong danh sách đang được xét có thể có nhiều thông tin, thông tin để dựa vào đó tìm kiếm thường gọi là khoá để tìm. Ví dụ: Khoá trong bài toán tìm các thí sinh trúng tuyển là điểm trung bình ba môn thi, khoá trong trong bài toán tìm các cán bộ về hưu là tổ hợp hai mục thông tin: giới tính và tuổi. Không mất tính tổng quát, chúng ta có thể xem xét bài toán tìm kiếm với khoá để tìm chỉ gồm một mục thông tin.

Có các thuật toán tìm kiếm sau đây:

- Tìm kiếm tuần tự (Linear searching).
- Tìm kiếm nhị phân (Binary searching).
- Tìm kiếm nhờ các bảng băm (Searching by using hash tables).
- Tìm kiếm theo chiều rộng (Breadth first search).
- Tìm kiếm theo chiều sâu (Deapth first search).
- Tìm kiếm Heuristic.

....

Người ta thường quan tâm đến tính hiệu quả của thuật toán tìm kiếm: thuật toán nào có tốc độ nhanh đương nhiên có hiệu quả cao hơn. Chúng ta hãy thử nghĩ xem: "Vì sao trong hàng chục triệu người đang truy nhập mạng Internet, thuật toán tìm kiếm nào mà chỉ trong vài giây đã kiểm tra xong có hay không có một User và một Password của một người vừa mới đăng nhập vào mạng?"

Sau đây, chúng ta chỉ xem xét các thuật toán tìm kiếm đơn giản trên một danh sách có thứ tự như mảng một chiều.

http://tieulun.hopto.org

4.1. Tìm kiếm tuần tự

Bài toán 1

Cho mảng a gồm n phần tử là các số thực. Hãy kiểm tra xem trong mảng có hay không một phần tử bằng x cho trước.

```
Bước 1: Khởi tạo Found:= False;
{ giả sử có x trong a}
Bước 2:
{Duyệt để kiểm tra và xác nhận nếu thấy x trong a}
For i:=1 to n do
If a[i]=x then Found:= True;
Bước 3: Nghiệm thu:
If Found then Writeln(' co ' ,x, ' trong mang a')
Else Writeln('khong co' ,x, ' trong mang a ');
```

Có thể cải tiến để giảm tối đa số lần duyệt như sau:

```
Bước 1: Khởi tạo i:=1;
Bước 2: Duyệt:
    While (i<=n) and (a[i] <> x) do inc(i);
Bước 3: Nghiệm thu:
 If i<=n then Writeln('co' ,x, ' trong mang a')
Else Writeln('khong co ' ,x, ' trong mang a');
```

Hoặc ta vận dụng lợi thế của lệnh ngắt break trong for:

```
Bước 1: a[n + 1]:= x;
Bước 2: Duyệt:
    For i:= 1 to n+1 do
    if (a[i] = x) then break;
Bước 3: Nghiệm thu:
 If i<=n then Writeln('co' ,x, ' trong mang a')
Else Writeln('khong co ' ,x, ' trong mang a');
```

Bài toán 2:

Cho mảng a gồm n phần tử là các số thực. Hãy chỉ ra vị trí của các phần tử trong mảng mà có giá trị bằng x cho trước.

http://tieulun.hopto.org

```
i:=1;
while (i<=n) and (a[i]<>x) do inc (i);
if i>n then writeln('khong co ' ,x, 'trong mang a ')
else
Begin
  Write(x, ' co trong mang a tai cac vi tri: ')
  for i:=1 to n do
  if a[i]=x then write(i, ','); Writeln;
End;
```

4.2. Tìm kiếm nhị phân

Bài toán 3:

Thực hiện yêu cầu của bài toán 1 trong trường hợp mảng a đã được sắp xếp và không mất tính tổng quát, ta có thể giả sử các phần tử của mảng được sắp xếp tăng dần.

- Tư tưởng thuật toán:

Ban đầu đoạn cần duyệt bắt đầu từ vị trí đầu d = 1 đến vị trí cuối c = n.

Thuật toán tìm kiếm x trên đoạn [d, c] thực hiện bởi quá trình lặp như sau:

Gọi m là vị trí chính giữa của đoạn tìm kiếm m = (d + c) div 2. Khi đó, vì dãy a tăng dần nên có 3 khả năng:

+ Nếu x > a[m] thì chứng tỏ x nằm ở bên phải vị trí m. Ta đánh dấu lại vị trí đầu của đoạn duyệt: d=m+1 và lặp lại việc tìm kiếm trên đoạn [d,c] mới.

+ Nếu x < a[m] thì chứng tỏ x nằm ở bên trái vị trí m. Ta đánh dấu lại vị trí cuối của đoạn duyệt: c=m-1 và lặp lại việc tìm kiếm trên đoạn [d,c] mới.

+ Nếu x = a[m] thì quá trình tìm kiếm x kết thúc.

Nếu không có x trong a thì thuật toán sẽ dừng khi d>c.

- Đoạn trình cốt lõi

```
Found:= False;  d:=1; c:=n;
while (d<=c) and not Found do
begin
      m:= (d+c) div 2;
      if x>a[m] then d:=m+1
      else if x <a[m] then c:=m-1
          else found:= true;
```


http://tieulun.hopto.org

```
end;
if found then writeln(x, ' co trong a ')
else writeln(x, 'khong co trong a');
```

4.3. Tìm phần tử lớn nhất, bé nhất

Bài toán 4:

Cho mảng a gồm n phần tử là các số thực. Hãy tìm giá trị lớn nhất của các phần tử trong mảng

- Vẫn áp dụng kỹ thuật 3 bước:

```
Bước 1: Khởi tạo max:= a[1]; { giả sử phần tử a[1] lớn nhất }
Bước 2: Duyệt để tối ưu dần
For i:=2 to n do
If max < a[i] then max:= a[i];
Bước 3: Nghiệm thu: Writeln('max = ', max);
```

Bài toán 5:

Cho mảng a gồm n phần tử là các số thực. Hãy tìm giá trị lớn nhất của các phần tử trong mảng và chỉ ra vị trí của các phần tử đó?

```
Giai đoạn 1: Tính max
max:= a[1];
For i:=2 to n do
If max < a[i] then max:= a[i];
Giai đoạn 2: in kết quả
    Writeln('max = ',max);
    Write('Tai cac vi tri: ');
    For i:=1 to n do
If a[i] = max then Write(i, #32); Writeln;
```

Chú ý: Không thể gộp hai giai đoạn trên thành một được.

5. Một số thuật toán sắp xếp đơn giản

Cũng như "tìm kiếm", sắp xếp là một vấn đề rất quan trọng trong lập trình vì nó có nhiều ứng dụng trong thực tế. Có thể kể ra đây một vài ví dụ để thấy được tầm quan trọng của thuật toán sắp xếp: Để chọn lựa học sinh giỏi người ta sẽ sắp xếp danh sách các học sinh theo thứ tự giảm dần (hoặc tăng dần) của điểm trung bình các môn học. Danh bạ điện thoại tuỳ vào từng hoàn cảnh có

http://tieulun.hopto.org

thể phải sắp xếp lại theo vần chữ cái, hoặc theo loại nhóm (group), hoặc theo vùng, miền. Khi nghiên cứu về từ vựng, đôi khi người ta muốn biết với một nhóm G các chữ cái và một tập S các từ tiếng Anh cho trước, thì liệu các từ trong H có cấu tạo với cùng một số lượng và loại chữ cái trong G hay không? Để trả lời câu hỏi đó, có thể ta sắp theo cùng chiều các ký tự trong từng từ rồi đối chiếu xem chúng có giống nhau không.

Có các thuật toán sắp xếp sau đây:

- Sắp xếp kiểu chọn trực tiếp (Selection sort).
- Sắp xếp kiểu nổi bọt (Buble Sort).
- Sắp xếp kiểu chèn trực tiếp (Insertion sort).
- Sắp xếp nhanh (Quick sort).
- Sắp xếp kiểu vun đống (Heap sort).
- Sắp xếp kiểu trộn (Merge sort).

...

Đương nhiên các thuật toán sắp xếp khác nhau có tốc độ thực hiện khác nhau và phụ thuộc vào độ lớn của danh sách cần sắp xếp, phụ thuộc vào thao tác sắp xếp tác động lên mảng thuộc bộ nhớ máy tính hay tác động lên tệp ở trên đĩa.

Chúng ta chỉ xem xét ba thuật toán đầu tiên là các thuật toán sắp xếp đơn giản.

5.1. Phát biểu bài toán

Cho mảng a gồm n số nguyên. Viết chương trình sắp xếp các phần tử của mảng theo thứ tự tăng dần.

5.2. Thuật toán sắp xếp kiểu "nổi bọt"

5.2.1. Tư tưởng thuật toán

Ta tưởng tượng rằng mảng được dựng đứng, phần tử thứ n nằm ở dưới đáy. Mỗi phần tử của mảng là một bọt khí có trọng lượng đúng bằng giá trị của nó. Bọt khí nhẹ hơn sẽ nổi lên trên, và bọt khí nặng sẽ chìm xuống. Quá trình nổi bọt diễn ra như sau: Ban đầu bọt khí nhẹ nhất sẽ dần dần được nổi lên trên cùng. Lần thứ 2, bọt khí nhẹ thứ nhì sẽ dần nổi lên ở vị trí thứ 2. Cứ tiếp tục quá trình như thế cho đến khi bọt khí nặng thứ nhì nằm ở vị trí sát đáy và như vậy mảng đã được sắp xếp.

http://tieulun.hopto.org

5.2.2. Mô tả thuật toán

Chúng ta có thể tuỳ chọn hình thức giả mã hoặc liệt kê từng bước để diễn tả thuật toán như sau:

a. Giả mã:

Algorithm Buble_Sort.
Function Sắp xếp n số $a_1,a_2,...,a_n$ theo thứ tự tăng dần.
Input n > 0 và n số $a_1,a_2,...,a_n$
Output $a_1,a_2,...,a_n$ được sắp xếp tăng dần
Format BubleSort($a_1,a_2,...,a_n$)
Method

```
1. For i=2 to n làm việc sau:
(* Chọn phần tử nhỏ nhất trong dãy a_i , .. , a_n để chuyển lên trước a_i *)
2.    For j=n down to i làm việc sau:
3.    If a[j] < a[j-1]
(* Phần tử ở vị trí j nhẹ hơn phần tử ở ngay phía trên *) then
4.        Swap(a[j], a[j-1]);
 (* Đổi giá trị của a[j] và a[j-1] cho nhau
   a[j] sẽ nổi lên, a[j-1] sẽ chìm xuống*)
```

End Buble_Sort.

b. Liệt kê từng bước:

Bước 1: Khởi tạo i = 2 (vị trí phần tử sát mặt nước)

Bước 2: Xét dãy a[i-1], a[i],... a[n], ta sẽ cho bọt khí nhẹ nhất của dãy này nổi lên trên cùng, tức là chiếm vị trí i-1 như sau:

Bước 3: Bắt đầu với j=n,

Bước 4: Ta kiểm tra xem phần tử ở vị trí j có nhẹ hơn phần tử ở ngay phía trên không (tức là kiểm tra có a[j] < a[j-1] không). Nếu đúng vậy thì a[j] sẽ nổi lên một 'nấc' và a[j-1] sẽ chìm xuống một nấc (tức là đổi giá trị của a[j] và a[j-1] cho nhau).

Bước 5: Nếu j=i thì thực hiện Bước 7.

Bước 6: Giảm j=j-1 và quay về Bước 4

Bước 7: Tăng i=i+1.

Lúc này các phần tử a[1]...a[i-2] đã được sắp xếp.

Bước 8: Nếu i<n thì quay về Bước 2.

Bước 9: Kết thúc.

http://tieulun.hopto.org

c. Dữ liệu kiểm thử với n = 5 và mảng a = { 6, 4, 5, 3, 1 }

		a[1]	a[2]	a[3]	a[4]	a[5]
i=2	j	6	4	5	3	1
	5	6	4	5	1	3
	4	6	4	1	5	3
	3	6	1	4	5	3
	2	1	6	4	5	3
i=3	5	1	6	4	3	5
	4	1	6	3	4	5
	3	1	3	6	4	5
i=4	5	1	3	6	4	5
	4	1	3	4	6	5
i=5	5	1	3	4	5	6

5.2.3. Đoạn chương trình

```
For i:=2 to n do
For j:=n down to i do
If a[j]<a[j-1] then
Begin
   tg:=a[j]; a[j]:=a[j-1]; a[j-1]:= tg;
End;
```

5.2.4. Đánh giá và cải tiến (đọc thêm)

a. Đánh giá số lần duyệt

i=2, j duyệt n-1 lần.

i=3, j duyệt n-2 lần.

Tổng quát với i<n, j duyệt (n-i-1) lần. Với i=n, j duyệt 1 lần.

Mặt khác, i sẽ duyệt (n-1) lần. Vậy tổng số lần duyệt của thuật toán bằng tổng số lần duyệt của j và bằng

(n-1) + (n-2) +... + 2 + 1 = (n-1)*(n-2) / 2 lần

http://tieulun.hopto.org

Ta nói rằng thuật toán có độ phức tạp xấp xỉ đa thức bậc 2: $f(n) = n^2$.

b. Thuật toán trên là nổi bọt từ dưới lên. Có thể có thuật toán hoàn toàn tương tự bằng cách "nổi bọt từ trên xuống".

```
for j:=n-1 down to 1 do
for i:=1 to j do
if a[i]>a[i+1] then
Begin
   tg:=a[i]; a[i]:=a[i+1]; a[i+1]:= tg;
End;
```

c. Trong trường hợp đặc biệt, toàn bộ mảng đã được sắp xếp thì thuật toán nổi bọt không xảy ra sự đổi chỗ lần nào, nhưng số lần duyệt vẫn không hề thay đổi. Tổng quát, trong một lần khởi tạo giá trị mới của i mà suốt quá trình j duyệt từ dưới lên i không xảy ra sự đổi chỗ lần nào thì mảng đã được sắp xếp và cần phải kết thúc quá trình duyệt. Đây chính là điểm cải tiến thứ nhất trong thuật toán sắp xếp nổi bọt:

```
i:=2;
Repeat
   stop:=True;
   For j:=n downto i do
 If a[j]<a[j-1] then
 Begin
    tg:=a[j]; a[j]:=a[j-1]; a[j-1]:= tg;
    stop:= False;
   End;
   i:=i+1;
Until stop or (i=n+1);
```

d. Có thể thu gọn quãng duyệt nhỏ hơn nữa bằng cách duyệt đồng thời từ dưới lên và từ trên xuống.

5.3. Thuật toán sắp xếp kiểu chọn trực tiếp

5.3.1. Tư tưởng thuật toán

Đầu tiên xét dãy n phần tử, ta duyệt trên dãy, bắt đầu từ phần tử thứ hai trở đi, cứ khi gặp phần tử nào nhỏ hơn phần tử đầu dãy thì ta đổi chỗ nó với phần

http://tieulun.hopto.org

tử đầu dãy. Kết thúc quá trình duyệt thì ta đã chuyển được phần tử bé nhất đặt lên đầu dãy.

Tiếp theo, xét dãy n-1 phần tử còn lại và làm tương tự như trên, nghĩa là ta duyệt trên dãy mới, bắt đầu từ phần tử thứ hai trở đi (tức là từ phần tử thứ ba so với dãy ban đầu), cứ khi gặp phần tử nào nhỏ hơn phần tử đầu dãy mới (tức là phần tử ở vị trí hai so với dãy ban đầu) thì ta đổi chỗ nó với phần tử đầu dãy mới. Kết thúc quá trình duyệt thì ta đã chuyển được phần tử bé nhất đặt lên đầu dãy mới (tức là trong dãy ban đầu, phần tử nhỏ thứ nhì đã được chuyển lên vị trí thứ hai). Tiếp theo, xét dãy n-2 phần tử còn lại và lại thực hiện công việc tương tự như trên.... Quá trình này kết thúc khi dãy còn lại chỉ còn một phần tử.

5.3.2. Mô tả thuật toán

a. Liệt kê từng bước:

Bước 1: Khởi tạo i=1
Bước 2: Khởi tạo j:=i+1;
Bước 3: Nếu a[j]<a[j] thì
Đổi giá trị của a[j] và a[i] cho nhau
Bước 4: Nếu j=n thì làm bước 6;
Bước 5: j:=j+1; làm bước 3;
Bước 6: Nếu i=n-1 thì làm Bước 8.
Bước 7: Tăng i:=i+1 và quay về Bước 2.
Bước 8: Kết thúc.

b. Giả mã:

Algorithm Selection_Sort.
Function Sắp xếp n số $a_1,a_2,....,a_n$ theo thứ tự tăng dần.
Input n > 0 và n số $a_1,a_2,...,a_n$.
Output $a_1,a_2,...,a_n$ được sắp xếp tăng dần.
Format SelectionSort($a_1,a_2,...,a_n$).
Method
1. For i=1 to n-1 làm các bước sau:
(* Xét dãy a_i , .. , a_n *)
2. For j=i+1 to n làm việc sau đây:
3. If a[j] < a[i] (* gặp phần tử nhỏ hơn phần tử đầu dãy*) then

http://tieulun.hopto.org

4. Swap(a[i], a[j])

(* Đổi chỗ phần tử nhỏ hơn này với phần tử đầu dãy *)

End Selection_Sort.

5.3.3. Đoạn chương trình

```
For i := 1 to n-1 do
For j := i + 1 to n do
If a[j] < a[i] then
Begin
begin
   tg:= a[i]; a[i]:=a[j]; a[j]:=tg;
end;
```

5.3.4. Đánh giá và cải tiến

a. Phương pháp cải tiến:

Đoạn trình trên tuy ngắn gọn nhưng lại thể hiện một thuật toán không tốt. Giả sử ta bắt đầu chuyển sang xét một dãy mới a[i], a[i+1],..., a[n]. Khi j duyệt trên dãy này, mỗi khi gặp một phần tử a[j] nhỏ hơn phần tử a[i] ở đầu dãy, ta không nên vội đổi chỗ a[j] với a[i] ngay mà chỉ cần lưu lại vị trí j của phần tử nhỏ hơn đó cho biến k. Khi j tiếp tục công việc duyệt trên dãy còn lại thì thao tác so sánh diễn ra giữa hai phần tử a[j] với phần tử a[k] chứ không phải là với a[i] (ban đầu khởi tạo k = i, ngay trước khi bắt đầu quá trình j duyệt trên dãy).

Như vậy khi j duyệt xong trên dãy mới thì k trỏ vào phần tử nhỏ nhất trong dãy đó. Nếu phần tử ở vị trí này không nằm ở đầu dãy (tức là nếu k<>i) thì ta tiến hành chuyển nó về đầu dãy mới bằng cách đổi chỗ a[i] và a[k] cho nhau.

Cải tiến trên tuy không làm giảm số lần duyệt và số lần so sánh nhưng lại làm giảm đáng kể số lần đổi chỗ.

b. Giả mã:

Algorithm Best_Selection_Sort.

Function Sắp xếp n số $a_1,a_2,...,a_n$ theo thứ tự tăng dần.

Input n > 0 và n số $a_1,a_2,...,a_n$

Output $a_1,a_2,...,a_n$ được sắp xếp tăng dần

Format BestSelectionSort($a_1,a_2,...,a_n$)

http://tieulun.hopto.org

Method

1. For i=1 to n-1 làm các bước sau:

(* Xét dãy a_i , .. , a_n *)

2. a. k = i; (* giả sử phần tử nhỏ nhất ở ngay đầu dãy*)

3. b. For j=i+1 to n làm việc sau đây:

4. If a[j] < a[i] (* gặp phần tử nhỏ hơn phần tử a[k]*) then

5. k := j; (* ghi nhận lại vị trí nhỏ hơn mới *)

6. c. If (k<>i) (* phần tử nhỏ nhất không ở đầu dãy*) then

7. Swap(a[i], a[k]);

(* Đổi chỗ phần tử nhỏ hơn này với phần tử đầu dãy *)

End Best_Selection_Sort.

c. Dữ liệu kiểm thử với n = 5 và mảng a = { 6,4,5,31 }

Bước	i	j chạy	k	a[1]	a[2]	a[3]	a[4]	a[5]
2	1	2->5	1	6	4	5	3	1
3			5					
4				1	4	5	3	6
5->6->2	2	3->5	2					
3			4					
4				1	3	5	4	6
5->6->2	3	4->5	3					
3			4					
4				1	3	4	5	6
5->6->2	4	5	5					
3			5					
4				1	3	4	5	6
5->6->7								

d. Đoạn chương trình:

```
For i:=1 to n-1 do
Begin
   k := i;
   for j:=i+1 to n do
```


http://tieulun.hopto.org

```
    if a[k]> a[j] then k:= j;
    if (k<>i) then
    begin
      tg:= a[i]; a[i]:=a[k]; a[k]:=tg;
    end;
end;
```

5.4. Thuật toán sắp xếp kiểu chèn trực tiếp (đọc thêm)

5.4.1. Tư tưởng thuật toán

Ban đầu có một phần tử được sắp xếp, đó là a[1].

Tổng quát với dãy i-1 phần tử được sắp xếp a[1],... , a[i-1] ta sẽ chèn a[i] vào vị trí thích hợp của dãy i-1 phần tử đã được sắp xếp này để được dãy mới có i phần tử được sắp xếp. Tiếp theo ta lại chèn phần tử a[i+1] vào dãy i phần tử đã được sắp xếp để được dãy mới có i+1 phần tử được sắp xếp... cứ tiếp tục cách làm như thế đến khi chèn xong phần tử a[n] vào vị trí thích hợp của dãy n-1 phần tử đã được sắp xếp thì ta được một mảng được sắp xếp hoàn chỉnh.

Như vậy trọng tâm của thuật toán là tìm cách đặt phần tử a[i] vào vị trí thích hợp của dãy i-1 phần tử a[1],...., a[i-1] đã được sắp xếp để sau khi chèn xong thì ta có dãy i phần tử được sắp xếp.

Ta khai báo thêm vị trí a[0] gọi là vị trí cầm canh và ban đầu gán a[0]:= a[i]

Với dãy a[1],...., a[i-1], ta dùng j để duyệt trên dãy này từ phải qua trái, nếu phần tử a[j] nào lớn hơn a[0] (tức là a[i]) thì tức là nó phải nằm bên phải a[i], nói cách khác, khi đó ta tịnh tiến a[j] sang phải một vị trí bởi lệnh a[j+1]:= a[j] và để lại vị trí "trống" j dành cơ hội đặt a[i] vào đó. Nói cách khác quá trình xét lùi dần sang trái là quá trình trình tịnh tiến các phần tử a[j] sang phải một vị trí. Đến một lúc nào đó có một phần tử a[j] < a[0] thì thôi quá trình tịnh tiến, vị trí trống sẽ để chèn a[i] vào đó.

5.4.2. Mô tả thuật toán

Algorithm Insertion_Sort.

Function Sắp xếp n số $a_1,a_2,...,a_n$ theo thứ tự tăng dần.

Input n > 0 và n số $a_1,a_2,...,a_n$

Output $a_1,a_2,...,a_n$ được sắp xếp tăng dần

http://tieulun.hopto.org

Format InsertionSort(a_1,a_2,....,a_n)

Method

1. For i=2 to n làm các bước sau:

{ Chèn a[i] vào vị trí thích hợp của dãy a[1],....,a[i-1] }

2. a. Gán a[0] = a[i];

3. b. Bắt đầu xem xét từ vị trí j = i - 1;

4. While a[j] > a[0] làm các bước sau đây:

5. i. a[j+1] = a[j]; (* tịnh tiến a[j] sang phải *)

6. ii. j = j - 1; (* xét tiếp phần tử bên trái *)

7. a[j+1] = a[0]; (* đặt a[i] vào vị trí bỏ trống hợp lý nhất *)

End Insertion_Sort.

5.4.3. Đoạn chương trình

```
for i:=2 to n do
begin
  { Chèn a[i] vào vị trí thích hợp của dãy a[1],....,a[i-1] }
   a[0]:= a[i];
   j:= i-1;
   while a[0] < a[j] do
   begin
      a[j+1]:= a[j];
      j:=j-1;
   end;
   a[j+1]:= a[0];
end;
```

Ý nghĩa của vị trí cầm canh a[0]

Khi tịnh tiến các phần tử bên trái a[i] sang phải thì rõ ràng giá trị của a[i] bị "đè mất" cho nên a[0] trước hết là để lưu lại giá trị này.

Quá trình tịnh tiến dừng lại khi có một a[j] mà a[j] < a[0] khi đó vị trí trống bên phải nó là a[j+1] được dành cho a[i], tức là kết thúc vòng lặp while thì a[j+1]:= a[0].

Trong trường hợp riêng, mọi phần tử a[1]...a[i-1] đều bé hơn a[i] thì vòng lặp vẫn dừng lại vì khi j giảm về 0 thì tất nhiên điều kiện a[0] <a[j] bị sai. Đây

http://tieulun.hopto.org

là tác dụng thứ 2 của vị trí cầm canh a[0] làm cho vòng lặp while luôn luôn phải dừng lại. Và trong trường hợp này vị trí trống là a[1] vì a[j+1] = a[0+1] để đặt a[i] vào đó.

6. Câu hỏi và bài tập

Bài 3.1

Một lớp có không quá 60 học sinh, thi môn TTLT.

1. Nhập vào sĩ số n của lớp, điểm thi của từng học sinh. Điểm tính theo thang điểm 10, chính xác đến 0.5.

2. Đếm xem trong lớp có bao nhiêu học sinh đạt khá giỏi (điểm >=6.5)?

3. Tìm và in lên màn hình điểm cao nhất, cụ thể là các học sinh có số hiệu nào?

4. Sắp xếp dãy điểm theo thứ tự giảm dần và in dãy điểm đó lên màn hình.

5. Hãy in số thứ tự của các học sinh cùng với điểm thi theo thứ tự tăng dần của điểm thi.

Bài 3.2

Viết chương trình:

1. Nhập vào n và n số nguyên (n<=20).

2. In lên màn hình các số nguyên tố trong dãy số vừa nhập.

3. In lên màn hình dãy đảo ngược của dãy đã nhập.

Bài 3.3

Viết chương trình:

1. Nhập vào mảng nguyên a có n phần tử (n<=20).

2. Cho biết các phần tử trong mảng a thỏa mãn tính chất nào sau đây:

a) Là dãy dừng.

b) Là dãy không giảm.

c) Là dãy không tăng.

d) Lập thành một cấp số cộng.

Chú ý: Một dãy tăng là một dãy không giảm nhưng ngược lại có thể không đúng.

Ví dụ:

{1, 3, 5} là một dãy tăng, đồng thời cũng là một dãy không giảm. Nhưng dãy { 1, 3, 3} là một dãy không giảm nhưng không phải dãy tăng.

http://tieulun.hopto.org

Bài 3.4

Có n loại mặt hàng (n<=26) ký hiệu là các chữ cái in hoa.

1. Nhập số loại mặt hàng và số lượng mặt hàng mỗi loại.

2. Tìm và in lên màn hình các loại mặt hàng có số lượng không ít hơn trung bình cộng số lượng của tất cả các mặt hàng.

3. Sắp xếp các loại mặt hàng theo thứ tự tăng dần về số lượng và in lên màn hình thành 2 dãy:

dãy 1: số hiệu các loại mặt hàng.

dãy 2: số lượng tương ứng của các loại mặt hàng đã được sắp.

Bài 3.5

In lên màn hình tam giác Pascal độ cao n (n<=20).

Tam giác Pascal là một hình ảnh được tạo bởi các con số trên các dòng, các con số của dòng thứ i là dãy các hệ số của dạng khai triển đa thức $(x+y)^i$.

Ví dụ sau đây là tam giác Pascal độ cao n = 5.

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

Bài 3.6

Nhập từ bàn phím một số nguyên có thể có 30 chữ số, kiểm tra xem số nguyên đó có thỏa mãn tính chất nào sau đây:

a) Là một số đối xứng.

b) Là một số chia hết cho 3.

Bài 3.7

Viết chương trình:

1. Nhập từ bàn phím số ngày công, chức vụ của n nhân viên trong một cơ quan

2. In lên màn hình lương của từng nhân viên. Trong đó, quy định lương một ngày công như bài 3.2.1 phần IV.

Bài 3.8

Viết chương trình:

Nhập một mảng gồm n số nguyên (n<=20) sao cho số được nhập sau không nhỏ hơn số đã nhập ngay trước đó.

Nhập tiếp một số nguyên x:

http://tieulun.hopto.org

Thực hiện việc chèn x vào trong mảng sao cho sau khi chèn thì mảng vẫn được sắp xếp. Tất nhiên trong thuật toán chèn không đưa x vào cuối dãy (hoặc đầu dãy) rồi áp dụng lại các thuật toán sắp xếp đã biết.

II. CẤU TRÚC DỮ LIỆU KIỂU MẢNG HAI CHIỀU

1. Định nghĩa, khai báo mảng hai chiều

1.1. Định nghĩa mảng hai chiều

Mảng hai chiều là một phân bố hai chiều gồm m hàng và n cột các phần tử có cùng một kiểu dữ liệu.

Có thể coi mảng 2 chiều là một mảng một chiều mà kiểu dữ liệu của mỗi một phần tử là kiểu mảng một chiều.

1.2. Khai báo mảng hai chiều

1.2.1. Khai báo trực tiếp

```
VAR
TênBiếnMảng: Array[Kiểu_cs_hàng, kiểu_cs_cột] OF Kiểu_phần_tử;
```

1.2.2. Khai báo gián tiếp

```
Type
Tên_kiểu_mảng: Array[ kiểu_cs_hàng, kiểu_cs_cột] OF Kiểu_phần_tử;
Var
Tên_biến_mảng: tên_kiểu_mảng;
```

Cũng giống như mảng một chiều, kiểu chỉ số hàng và kiểu chỉ số cột của mảng hai chiều là các kiểu dữ liệu đoạn con và xác định vị trí của các phần tử của mảng theo hàng và theo cột.

Kiểu phần tử của mảng quy định kiểu dữ liệu của các phần tử trong mảng. Kiểu phần tử của mảng có thể là kiểu dữ liệu chuẩn hoặc phức tạp hơn là kiểu dữ liệu có cấu trúc đã được định nghĩa bởi chính người lập trình.

Các phần tử của mảng hai chiều cũng được truy nhập thông qua tên biến mảng và tọa độ của nó trong mảng, nghĩa là chỉ ra vị trí hàng và vị trí cột của nó.

```
Tên_biến_mảng [ vị_trí_hàng, vị_trí_cột];
```

1.3. Ví dụ về khai báo mảng hai chiều

Các ví dụ sau đây không những minh họa cách khai báo mảng hai chiều mà còn gợi ý *cách biểu diễn vấn đề* được nêu ra trong bài toán (chưa cần quan tâm đến yêu cầu của bài toán).

http://tieulun.hopto.org

Khi giải bài toán trên máy tính, ta chưa quan tâm ngay đến việc giải quyết bài toán đó như thế nào. Việc đầu tiên là tìm cách biểu diễn vấn đề (các đối tượng) nêu trong bài toán. Kỹ thuật thể hiện thuật toán (thậm chí, đôi khi, kể cả chính thuật toán) phụ thuộc rất nhiều vào cách tổ chức dữ liệu cho bài toán.

Ví dụ 1

Một lớp có m học sinh, ký hiệu từ 1,2,..., m. (1<=m <= 60) thi n môn ký hiệu là 1, 2,..., n (1<= n <= 20).

Xét việc tổ chức dữ liệu biểu thị bảng điểm các môn của học sinh lớp đó.

```
Const maxM=60; maxN=20;
Type Bangdiem = Array[1..maxM,1..maxN] of Real;
Var
A: Bangdiem;
m,n, i,j : integer;
```

Ví dụ cụ thể với m=3. n = 4, bảng điểm A như sau:

	1	2	3	4
1	6.5	6	5.5	6.5
2	4	9	8	8
3	7	6.5	6	7

Như vậy A[1,3]=5.5 biểu thị điểm môn 3 của học sinh 1 là 5.5. Tổng quát, A[i,j] là điểm của học sinh i thi môn j. Các hàng i biểu thị học sinh, các cột j biểu thị môn, i=1,2,..,m; j = 1,2,...,n.

Ví dụ 2

Có m mặt hàng ký hiệu là các chữ cái in hoa 'A','B',.... (m<=26). Người ta quan tâm đến doanh thu của các mặt hàng này trong vòng các năm gần đây, từ 1990 đến 2000.

Xét việc chức biểu diễn doanh thu của các mặt hàng:

```
Var
B: Array['A'..'Z',0..10] of longint;
m,n: integer;
i: char;
j: integer;
```


http://tieulun.hopto.org

Ví dụ cụ thể với m=4, n =4, các số liệu sau đây giả sử đơn vị là triệu đồng, với n=4 ta có các năm 1990, 1991, 1992, 1993.

	0	1	2	3
'A'	50	52	54	55
'B'	40	38	41	35
'C'	20	26	30	21
'D'	40	35	30	24

Như vậy doanh thu của mặt hàng 'B' năm 1992 là:
B['B',2]=41 triệu đồng. Nếu i:='B', j=2 ta có thể in lên màn hình như sau:
Writeln('B[' ,i, ',' , j+1990, '] = ', B[i,j]);

Ví dụ 3

Một khu triển lãm cần một bảng trực trong n ngày, mỗi ngày có m ca trực. Bảng phân công có dạng như ví dụ sau với m=3 ca, n=4 ngày:

	1	2	3	4
1	lan	ha	binh	hanh
2	hanh	lan	lien	hoa
3	lien	binh	lan	lan

Vậy ta có thể khai báo:

```
Const
maxn = 30; maxm = 4;
Var
C: array[1..maxn, 1..maxm] of string;
m,n: integer;
i,j: integer;
```

2. Vào/ra mảng hai chiều

Xét khai báo sau đây:

```
Const
maxM =20;
maxN = 30;
Var
A: Array[1..maxM, 1..maxN] of real;
m,n: integer;
i,j: integer;
```


http://tieulun.hopto.org

2.1. Mẫu nhập mảng hai chiều từ bàn phím

2.1.1. Nhập mảng hai chiều không có điều kiện

```
{Bước 1}
Repeat
   Write('nhap so hang m = '); Readln(m);
Until (1<=m) and (m<=maxM);
Repeat
   Write('nhap so cot n = '); Readln(n);
Until (1<=n) and (n<=maxN);
{Bước 2}
For i:=1 to m do
For j:=1 to n do
Begin
 Write('A[' ,i, ',' ,j, ']= '); Readln(A[i,j]);
End;
```

2.1.2. Nhập mảng hai chiều có điều kiện

Nếu A biểu diễn bảng điểm thì 0<=A[i,j] <=10, khi đó trong mẫu (a). Bước 2 phải sửa lại là:

```
For i:=1 to m do
For j:=1 to n do
Repeat
 Write('A[' ,i, ',' ,j, ']= '); Readln(A[i,j]);
Until (A[i,j]>=0) and (A[i,j]<=10);
```

2.2. Mẫu in mảng hai chiều lên màn hình

```
For i:=1 to m do
Begin
   For j:=1 to n do write(A[i,j]:5:1);
   Writeln;
End;
```

3. Duyệt mảng hai chiều và kỹ thuật 3 bước trong lập trình

3.1. Duyệt trên một hàng

Trong một mảng hai chiều A (m hàng và n cột), khi cần quan tâm đến một hàng i_0 cố định, để duyệt trên hàng đó, ta có thể áp dụng mẫu sau đây:

http://tieulun.hopto.org

Bước 1: Khởi tạo (nếu có)
Bước 2: Duyệt trên hàng i_0
For j:=1 to n do
[if <A[i_0,j] thỏa mãn dk > then] < xử lý A[i_0, j]>
Bước 3: Nghiệm thu sau duyệt.

Ví dụ 1

Viết đoạn chương trình tính điểm trung bình của học sinh cuối cùng trong lớp.

```
s:=0;
For j:=1 to n do s:= s + a[m,j];
Writeln('diem trung binh hs thu ',m, ' la: ', s/n:0:1);
```

3.2. Duyệt trên một cột

Tương tự như khi duyệt trên một hàng cố định, khi cần quan tâm đến một cột j_0 cố định, để duyệt trên cột đó, ta có thể áp dụng mẫu sau đây:

Bước 1: Khởi tạo (nếu có)
Bước 2: Duyệt trên cột j_0
For i:=1 to m do
[if <A[i j_0] thỏa mãn dk > then] < xử lý A[i, j_0] >
Bước 3: Nghiệm thu sau duyệt.

Ví dụ 2

Viết đoạn chương trình cho biết môn thi đầu tiên điểm cao nhất là bao nhiêu.

```
max:= 0;
for i:=1 to m do
if max < a[i,1] then max:= a[i,1];
writeln('mon 1, diem cao nhat la: ', max:5:1);
```

3.3. Duyệt toàn bộ mảng

Để duyệt tất cả các phần tử trong một mảng hai chiều A gồm m hàng, n cột, ta lần lượt cố định các hàng i = 1,2,..,m. Với mỗi hàng i cố định đó, ta áp dụng mẫu duyệt cố định trên một hàng. Cách duyệt như thế gọi là phương pháp duyệt ưu tiên theo hàng.

http://tieulun.hopto.org

Tương tự, nếu ta lần lượt cố định từng cột j =1,2,...,n và với mỗi cột j cố định ấy, ta áp dụng mẫu duyệt cố định trên một cột. Cách duyệt này gọi là phương pháp duyệt ưu tiên theo cột.

Sau đây là duyệt mảng ưu tiên theo hàng trình bày theo kỹ thuật 3 bước:

```
Bước 1: Khởi tạo tổng thể (nếu có)
Bước 2: Duyệt toàn bộ
For i:=1 to m do
begin
  2.a. Khởi tạo cục bộ (nếu có)
  2.b. Duyệt trên hàng i
     for j:=1 to n do
     [ if <a[i,j] thỏa mãn dk> then]  < xử lý a[i,j] >;
  2.c. Nghiệm thu cục bộ;
end;
Bước 3: Nghiệm thu tổng thể;
```

Ví dụ 3

Hãy tính và in lên màn hình điểm trung bình cho từng học sinh và cho biết điểm trung bình thấp nhất là bao nhiêu?

```
min:= 10.0;
for i:=1 to m do
begin
   tb:= 0;
   for j:=1 to n do tb:= tb + a[i,j]; tb:= tb/n;
   writeln('hs thu ' ,i, ' tbm = ', tb: 5:1);
   if min > tb then min:= tb;
end;
writeln('diem trung binh thap nhat la: ', min:5:1);
```

4. Câu hỏi và bài tập

Bài 3.9

Có m điểm phát (cấp phát hàng) ký hiệu là 1,2,...,m và n điểm thu (tiêu thụ hàng), ký hiệu là 1,2,...,n (m<=20, n<=30). Người ta cần quan tâm đến số lượng của một loại hàng cố định nào đó trong việc thu và phát. Viết chương trình:

http://tieulun.hopto.org

1. Nhập từ bàn phím m, n và số lượng hàng của m điểm phát đến từng địa điểm thu.

2. Cho biết chênh lệch về tổng số lượng hàng cấp phát giữa điểm phát đầu tiên và điểm phát cuối cùng.

3. Cho biết điểm thu nào tiêu thụ được nhiều hàng nhất, cụ thể bao nhiêu?

Bài 3.10

Một nhóm gồm m sinh viên (m<=25) ký hiệu là 1,2,..,m, tham gia thi tìm hiểu n đề tài (n<20), ký hiệu là các chữ cái in thường'a','b',... Ban tổ chức đã chấm điểm các đề tài cho các sinh viên theo thang điểm 20, chính xác đến 0.5. Viết chương trình:

1. Nhập m,n và điểm của các sinh viên tham gia các đề tài. Quy định nhập điểm -1 cho một đề tài nếu sinh viên đó không đăng ký thi đề tài đó.

2. Cho biết đề tài nào có ít sinh viên từ chối tham gia nhất.

3. Điểm cao nhất là bao nhiêu, đó là những sinh viên nào và đề tài nào.

4. Tính và in lên màn hình điểm trung bình của từng sinh viên.

Bài 3.11

Để chuẩn bị cho SeaGames lần thứ 22, Uỷ ban Olympic quốc gia có dự án xây dựng tổ hợp thể thao gồm N hạng mục công trình: sân vận động, nhà thi đấu, khu công viên, làng Olympic, trung tâm báo chí, đường vành đai ... Có M nhà thầu được tham dự đấu thầu. Nhà thầu thứ i dự trù kinh phí cho hạng mục thứ j là C[i,j] triệu đồng.

Viết chương trình:

1) Nhập giá trị cho m, n và mảng C (các số nguyên dương). (m<=10, n <=20).

2) Nhập vào tên của N hạng mục công trình và tên của M nhà thầu.

3) In lên màn hình tên của các hạng mục công trình cùng với tên của nhà thầu tương ứng trúng thầu. Người được trúng thầu là người đệ trình một dự trù kinh phí nhỏ nhất.

4) In lên màn hình tổng kinh phí hợp đồng với các nhà thầu.

Bài 3.12*

Một mảng vuông A[n x n] có thể biểu diễn điểm của n đội bóng trong một bảng thi đấu vòng tròn với quy ước:

- A[i,j] = 0 nếu đội i thua đội j
- A[i,j] = 1 nếu đội i hoà đội j
- A[i,j] = 2 nếu đội i thắng đội j

(1<=i,j<=n)

http://tieulun.hopto.org

Nhận thấy, A là một mảng đối ngẫu, nghĩa là:

A[j,i]=2-A[i,j].

Như vậy chỉ cần biết giá trị của mảng A phần tam giác bên trên đường chéo chính thì sẽ suy ra được giá trị của mảng A phần tam giác bên dưới đường chéo chính.

Để chặt chẽ, ta quy ước A[i,j] = -1 nếu i=j.

Viết chương trình

1) Nhập từ bàn phím số nguyên n (n<=15) và giá trị của nửa mảng A - phần tam giác bên trên đường chéo chính. Nghĩa là với mỗi đội i=1,2,...,n-1 , ta sẽ nhập điểm của đội i thi đấu với n-i đội còn lại là các đội: j = i+1,...,n.

2) Chiếu lên màn hình toàn bộ bảng điểm A.

3) Hãy cho biết những đội nào thắng nhiều nhất? Cụ thể bao nhiêu điểm?

4) Hãy cho biết tổng số lần thắng, tổng số lần thua của một đội bóng k nào đó, k nhập từ bàn phím?

5*) Hãy in lên màn hình bảng xếp hạng của các đội gồm 2 cột số hiệu đội và tổng điểm theo thứ tự giảm dần của tổng điểm.

Bài 3.13*

Hình vuông kỳ ảo bậc n được định nghĩa là một mảng 2 chiều kích thước n x n (còn gọi là ma trận vuông cấp n) sao cho:

- Chứa đủ n^2 số tự nhiên đầu tiên,
- Các tổng sau đây bằng nhau: Tổng các số trên từng hàng, tổng các số trên từng cột, tổng các số trên đường chéo chính, tổng các số trên đường chéo phụ.

Viết chương trình nhập số tự nhiên lẻ n, in lên màn hình một hình vuông kỳ ảo bậc lẻ đó.

Ví dụ dưới đây là 2 hình vuông kỳ ảo bậc 3 và bậc 5:

8	1	6
3	5	7
4	9	2

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

http://tieulun.hopto.org

III. DỮ LIỆU KIỂU XÂU KÝ TỰ

1. Định nghĩa, khai báo xâu ký tự

1.1. Định nghĩa xâu ký tự

Xâu ký tự là một kiểu dữ liệu nửa cấu trúc biểu diễn một danh sách có thứ tự các ký tự trong bảng ký tự của ngôn ngữ đang sử dụng.

Xâu ký tự có tính cấu trúc vì nó có thể coi là một mảng một chiều các phần tử kiểu char, song nó cũng phi cấu trúc ở chỗ có thể đọc hoặc ghi tất cả một biến kiểu xâu ký tự.

1.2. Khai báo biến xâu ký tự

VAR tên_biến_xâu: STRING[n];

Trong đó n phải là một số cụ thể $1 \leq n \leq 255$ biểu thị độ dài (số ký tự) tối đa của xâu ký tự chứa trong biến đó.

Nếu n = 255 thì có thể viết nguyên từ khóa STRING là đủ.

Ví dụ:

```
VAR
   hoten: string[26];
   diachi: string[100];
   qtct   : string;
```

2. Các phép toán trên xâu ký tự

Xét khai báo

VAR x,y,z: string;

2.1. Phép gán

Có thể khởi tạo một chuỗi ký tự (hằng xâu ký tự) cho một biến xâu hoặc gán hai xâu ký tự cho nhau.

Ví dụ:

x := 'Nguyen Thu Ha';

y := x;

Các ký tự trong xâu có thể truy nhập thông qua vị trí giống như mảng một chiều.

Ví dụ:

Nếu

y := 'Le Thi Ha';

http://tieulun.hopto.org

Thì ta có y[6] = 'i'

2.2. Phép so sánh

Phép so sánh hai xâu ký tự dựa vào phép so sánh mã ASCCII của hai ký tự khác nhau đầu tiên tính từ trái sang phải trên hai xâu đó.

Ví dụ:

x:= 'HONG HA';

y:= x;

Khi đó x=y

x:= 'LAM' :

y:= 'LAN';

Khi đó x < y vì M đứng trước N nên ord('M') < ord('N').

2.3. Phép cộng xâu

Phép cộng xâu như một phép ghép xâu.

Phép cộng xâu không có tính chất giao hoán.

Ví dụ:

x:='THU ';

y:='HOAI ';

nếu z := x+y; thì z = 'THU HOAI ';

nếu z := y+x; thì z ='HOAI THU ';

2.4. Duyệt trên xâu ký tự

Nếu xâu s có độ dài n thì phép duyệt một lần trên xâu s tổng quát như sau:

```
For i:=1 to n do
if <s[i] thỏa mãn đk> then < xử lý s[i] >;
```

3. Các hàm và thủ tục trên xâu ký tự

Hàm	Thủ tục
Length(s)	delete(s,i,n)
Pos(x,s)	insert(x,s,before_i)
Copy(s,i)	val(s,n,code)
	str(n,s):

3.1. Các hàm trên xâu ký tự

Hàm Length(s) cho biết độ dài hay số lượng ký tự của xâu s.

Ví dụ:

s := 'Ngo Thu Nga';

http://tieulun.hopto.org

vậy length(s) = 11

Hàm pos(x,s) cho biết vị trí xuất hiện đầu tiên của xâu x trong xâu s. Nếu không có x trong s thì hàm cho kết quả bằng 0.

Ví dụ:

pos('Thu', s) cho giá trị 5.

Nếu s:= 'mnabxyabab'; x:='ab';

Thì lệnh pos(x,s) dù thực hiện bao nhiêu lần cũng chỉ cho kết quả là 3.

Hàm copy(s,i,n) cho giá trị là một xâu ký tự con của xâu s, gồm n ký tự, tính từ vị trí thứ i trên xâu s.

Ví dụ:

copy('le hang nga',3,4) cho kết quả là 'hang'.

Nếu s := 'ab cd mn pq';

Thì hàm copy(s,6,length(s)) cho kết quả là xâu ' mn pq'.

3.2. Các thủ tục trên xâu ký tự

Thủ tục delete(s,i, n) xóa đi n ký tự của xâu s tính từ vị trí i.

Ví dụ:

s:= 'le thi phuong thuy';

delete(s,4,4);

cho kết quả s = 'le phuong thuy'.

Thủ tục insert(x,s,i) sẽ chèn xâu x vào trong xâu s và đặt x trước vị trí thứ i trong xâu s.

Ví dụ:

s:='';

insert('do mi hang',s,1);

cho kết quả s = 'do mi hang'.

Tiếp theo: insert('nguyen ',s,1); cho kết quả s ='nguyen do mi hang'.

Tiếp theo: insert('thi ',s,10); cho kết quả s='nguyen do thi mi hang'.

Thủ tục Val(s,n,code) đổi xâu s thành số n. Nếu phép đổi đúng thì code bằng 0, nếu phép đổi sai thì code > 0, hơn nữa giá trị của code là vị trí gây lỗi (không phải là chữ số) trong xâu s.

Ví dụ:

với s = '1234' thì val(s,n,c) cho kết quả:

http://tieulun.hopto.org

n=1234 và c = 0

với s = '123a4' thì val(s,n,c) cho kết quả n không xác định còn c= 4.

Thủ tục str(n,s) đổi số n thành xâu ký tự s.

Ví dụ:

n=3324 thì str(n,s) cho kết quả s='3324'.

4. Một số thuật toán - bài toán xử lý xâu ký tự

4.1. Bài toán: Đếm xâu con

Viết chương trình đếm xem trong câu 'nhan dan viet nam can cu lao dong anh dung danh giac' vần 'an' xuất hiện bao nhiêu lần.

4.1.1. Ý tưởng thuật toán

Tổng quát cho 2 xâu x,s. Ta sẽ đếm xem x xuất hiện trong s bao nhiêu lần.

Ví dụ:

x:='ab' ; Đặt Lx = length(x) = 2;

s:='abcabdabm'; Ta dễ thấy x xuất hiện 3 lần trong s.

Gọi d là biến đếm số lần xuất hiện của x trong s.

Ban đầu khởi tạo d:= 0;

Ta cho i nhìn vào các ký tự của s bắt đầu từ vị trí 1. Tại vị trí i=1, ta trích ra một xâu con có độ dài 2 và so sánh với x, và ta kiểm tra thấy copy(s,i,2) = x, vậy x xuất hiện lần thứ nhất trong s, ta tăng 1 cho biến đếm d. Tiếp theo cho i=2 và lại trích ra từ xâu s một xâu con gồm 2 ký tự bắt đầu từ vị trí 2 của s, kết quả kiểm tra cho thấy copy(s,i,2) <> x, ta không tăng giá trị cho d. Tiếp theo ta lại tăng i=3, rồi lại kiểm tra đẳng thức copy(s,i,2) = x đúng hay sai để quyết định có cộng 1 vào d hay không... cứ tiếp tục quá trình như thế đến khi i=length(s)-1.

Tổng quát, với mỗi giá trị của i=1,2,....length(s)-Lx-1, ta tiến hành kiểm tra nếu copy(s,i,Lx)=x thì tăng 1 cho biến đếm d. Kết thúc quá trình duyệt thì d chính là số lần xuất hiện xâu x trong xâu s.

4.1.2. Bảng mô phỏng Với x:='ab' ; s:='abcabdabm

	i	copy(s,i,lx)	copy(s,i,lx)=x	d
s =				0
a	1	ab	true	1
b	2	bc	false	
c	3	ca	false	
a	4	ab	true	2

http://tieulun.hopto.org

b	5	bd	false	
d	6	da	false	
a	7	ab	true	3
b	8	b	false	
m		m	false	

4.1.3. Chương trình

DEMVAN.PAS

```
Var
x,s: string;
lx,i,d: byte;
Begin
write('s = '); readln(s);
write('x = '); readln(x);
lx:= length(x); d:=0;
for i:=1 to length(s)-lx+1 do if copy(s,i,lx)=x then inc(d);
Writeln(' " ' ,x, ' " xuat hien trong " ' ,s, ' " so lan = ' ,d);
Readln;
End.
```

4.2. Bài toán: Xóa dấu cách vô nghĩa

Một từ là một dãy ký tự liên tục không chứa dấu cách bên trong. Một xâu 'sạch' là một xâu mà giữa các từ có đúng một dấu cách và không có dấu cách nào ở đầu xâu và cuối xâu.

Viết chương trình nhập vào một xâu S và làm sạch xâu này.

4.2.1. Ý tưởng thuật toán

Vì dấu cách thì không nhìn thấy được, cho nên, để cho dễ nhìn ta thay dấu cách bằng ký tự *

Ví dụ, xâu S dưới đây không sạch:

S = '**Ngo ****Thi **Thu***Nga***'

Bước 1:

Lặp quá trình sau khi trong S không có hai dấu cách liền nhau: nếu còn hai dấu cách liền nhau thì xóa khỏi s một dấu cách tại vị trí hai dấu cách

http://tieulun.hopto.org

liền nhau đó. Như vậy kết thúc bước 1 thì giữa các từ trong S có đúng mộ
dấu cách.

Ta có thể khai báo một hằng space =#32 để tiện dùng trong chương trình.

```
k:=pos(space+space, s);
while k>0 do
begin
   delete(s,k,1);
   k:=pos(space+space, s);
end;
```

Kết thúc bước 1 thì s = '*Ngo *Thi *Thu*Nga*'

Bước 2:

Xóa một dấu cách ở đầu (nếu có) và một dấu cách ở cuối S (nếu có)

```
if s[1] = space then delete(s,1,1);
if s[length(s)]=space then delete(s,length(s),1);
```

4.2.2. Chương trình

CLEAN.PAS

```
const
   space=#32;
var
   s:string;
   k:byte;
Begin
   write('s = ');
   Readln(s);
   writeln('before clean s = ',s);
   k:=pos(space+space,s);
   while k >0 do
   begin
        delete(s,k,1);
         k:=pos(space+space, s);
   end;
   if s[1] = space then delete(s,1,1);
```


http://tieulun.hopto.org

```
        if s[length(s)]=space then delete(s,length(s),1);
        writeln('after clean s = ',s);
        readln;
    End.
```

4.3. Bài toán: Sửa lỗi cú pháp về dấu ngăn cách trong văn bản

Trong một văn bản không bị lỗi về dấu ngăn cách thì sau các dấu phẩy và sau dấu chấm có đúng một dấu cách và trước các dấu này không có dấu cách nào.

Ví dụ văn bản sau đây là bị lỗi: (để dễ nhìn ta thay dấu mỗi một dấu cách bằng một ký tự * - khi viết chương trình học sinh phải nhập đúng dấu cách)

Troi*mua**,khong*to.***khong*nho***.*Mua*keo*dai**,**day dut*,dai*dang*.

Mai*hien*nuoc*chay*rong*rong.Cay*coi*dung*ban khoan,phan*van.

Cần phải sửa lại để không lỗi như sau:

Troi*mua,*khong*to.*khong*nho.*Mua*keo*dai,*day dut,*dai*dang.

*Mai*hien*nuoc*chay*rong*rong.*Cay*coi*dung*ban khoan,*phan*van.

Viết chương trình nhập vào một chuỗi ký tự biểu diễn một đoạn văn ngắn đã chính xác về mặt ngữ pháp và không quá 255 ký tự. Hãy sửa lại văn bản trên để không bị lỗi về dấu ngăn cách.

4.3.1. Ý tưởng thuật toán

Ta sẽ sửa lỗi cho từng loại dấu. Cách sửa lỗi cho các dấu là như nhau, vì vậy, ta chỉ cần trình bày thuật toán sửa lỗi dấu phẩy:

Để đơn giản, xét đoạn văn ngắn sau đây:

s='ab,cd**,**mn***,*pq,*uv*,xy.'

Định nghĩa các hằng ký tự sperator = ',' : Space = #32:

Bước 1: Thêm vào sau mỗi dấu phẩy một dấu cách.

*{Kết quả s='ab,*cd**,***mn***,**pq,**uv*,*xy.'}*

Bước 2: Lặp quá trình xoá đi một dấu cách trong hai dấu cách liên tiếp nếu còn tìm thấy hai dấu cách liên tiếp.

*{Kết quả s='ab,*cd*,*mn*,*pq,*uv*,*xy.'}*

Bước 3: Xóa một dấu cách trước dấu phẩy (nếu có)

*{Kết quả s='ab,*cd,*mn,*pq,*uv,*xy.'}*

http://tieulun.hopto.org

4.3.2. Chương trình

SYTAX.PAS

```
const
   space   = #32;
   sperator =',';
   point    ='.';
Var
   s: string;
   i: byte;
Begin
   write('s = '); Readln(s);
   { buoc 1}
   i:=0;
   Repeat
        i:=i+1;
        if s[i]= sperator then  Insert(space,s,i+1);
        if s[i]= point then  Insert(space,s,i+1);
   Until i=length(s)-1;
   {buoc 2}
   Repeat
        i:=pos(space+space,s);
        if i>0 then delete(s,i,1);
   Until i=0;
   {buoc 3}
   i:=0;
   Repeat
        i:=i+1;
        if (s[i]=space)and(s[i+1]=sperator) then Delete(s,i,1);
        if (s[i]=space)and(s[i+1]=point) then Delete(s,i,1);
   Until i=length(s)-1;
   Writeln('Valid s = ',s);
   Readln;
End.
```


http://tieulun.hopto.org

4.4. Bài toán: Thống kê số từ

Cho một câu (xâu) S nhập từ bàn phím. Hãy in các thông tin sau lên màn hình:

1. Số từ.
2. Số từ độ dài 3.
3. Một từ dài nhất.

4.4.1. Ý tưởng thuật toán

Bài toán này có nhiều hơn một cách giải. Sau đây ta sẽ xem xét một cách.

Xét cả ba yêu cầu ta thấy trước hết cần làm sạch xâu S như thuật toán bài toán xoá dấu cách vô nghĩa nhưng vẫn để lại hai dấu cách ở hai đầu xâu S, nếu không có các dấu cách này thì ta tự thêm vào. Việc thêm vào hai dấu cách ở hai đầu xâu S là để thuận tiện giải quyết yêu cầu 2 và yêu cầu 3 của đầu bài: lúc đó ta sẽ dùng hai biến i và j để lần lượt trỏ vào 2 đầu từng từ trong xâu S mà chúng ta cần phải tính độ dài từ đó. Lưu ý rằng ký tự hai đầu một từ sẽ là dấu cách, nói cách khác nếu i và j nhìn vào hai dấu cách thì chứng tỏ giữa i và j là một từ.

```
s:=space+s+space;
repeat
   i:=pos(space+space,s);
   if i>0 then delete(s,i,1);
until i=0;
```

Yêu cầu 1) Đếm từ

Giống bài toán trồng cây 2 đầu đường, thấy rằng số từ trong S là số lượng dấu cách trừ đi 1.

```
st:=0;
for i:=1 to lengh(s) do
if s[i]=space then inc(st);
writeln('so tu: ',st-1);
```

Yêu cầu 2, yêu cầu 3) Đếm số từ có độ dài 3 (có 3 ký tự) và tìm từ dài nhất:

Cho i nhìn vào dấu cách tiếp theo, j nhìn vào dấu cách ngay phía sau i.

Độ dài của từ đang xét là k = j-i-1.

http://tieulun.hopto.org

Nếu độ dài của từ: k = 3 thì ta đếm cho biến st3.

Nếu độ dài từ đang xét k < max thì gán lại max = k và lưu lại từ dài hơn đó. (max là độ dài từ dài nhất. được khởi tạo "ít nhất" bằng 0)

```
st3:= 0; max:= 0;
i:=1;
Repeat
   j:=i+1; while (j<length(s)) and (s[j]<>space) do inc(j);
    if j-i-1=3 then inc(st3);
  if j-i-1 > max then
    begin
            max:= j-i-1;
       x:=copy(s,i+1,max);
    end;
    i:=j;
Until j=length(s);
```

4.4.2. Chương trình:

PROCESS.PAS

```
const
   space = #32;
var
   s: string;
   ls, i,j:byte;
   st,st3: byte;
   x: string;
begin
   write('nhap xau s: '); readln(s);
   s:=space+s+space;
   repeat
        i:=pos(space+space,s);
        if i>0 then delete(s,i,1);
   until i=0;
```


http://tieulun.hopto.org

```
ls:=length(s);
st:=0;
for i:=1 to ls do
if s[i]=space then inc(st);
writeln('so tu: ',st-1);

st3:= 0; max:= 0;
i:=1;
Repeat
      j:=i+1; while (j<length(s)) and (s[j]<>space) do inc(j);
      if j-i-1=3 then inc(st3);
      if j-i-1 > max then
      begin
                  max:= j-i-1;
                  x:=copy(s,i+1,max);
      end;
Until j>=length(s);

   write('tu dai nhat ',x);
   readln;
end.
```

4.5. Bài toán: Hai từ anagram

Hai từ x và y gọi là anagram với nhau nếu mỗi ký tự của từ này cũng có mặt trong từ kia và hơn nữa số lượng từng loại ký tự xuất hiện trong hai từ là bằng nhau.

Ví dụ, trên mỗi dòng sau đây là danh sách các từ anagram của nhau:

read, dear, dare

tea, eat, ate,

bad, dab

bale, able

Trên mỗi dòng sau, các từ không là anagram của nhau:

http://tieulun.hopto.org

feel, fell (cảm thấy, da lông)

kitchen, chicken (nhà bếp, thịt gà)

Viết chương trình kiểm tra 2 từ x và y cho trước có là anagram của nhau hay không.

Gợi ý:

Ta có thể xem mỗi từ như một mảng các ký tự (xâu ký tự có thể được xử lý như một mảng các ký tự).

Ta lần lượt sắp xếp cùng chiều các ký tự trong mảng x rồi mảng y. Nếu sau khi sắp xếp mà x = y thì hai từ x, y ban đầu là anagram của nhau.

Chú ý rằng trong thuật toán sắp xếp, việc so sánh các ký tự là hoàn toàn chấp nhận vì đó là phép so sánh hai ký tự theo mã ASCII của chúng.

5. Câu hỏi và bài tập

Bài 3.14

Giả sử có hằng xâu ký tự :

s = 'Hoang Anh, Hong Anh, Lan Anh'

Hãy cho biết giá trị của k bằng bao nhiêu sau khi lệnh:

k:=POS('Anh',s); thực hiện 3 lần?

Bài 3.15

Xâu ký tự gồm các thủ tục và hàm nào được giới thiệu trong bài học? Qua việc sử dụng các hàm và thủ tục trên xâu ký tự hãy cho biết sự khác nhau giữa hàm và thủ tục về cách sử dụng?

Bài 3.16

1) Phép cộng xâu có tính chất giao hoán không?

2) Phép so sánh 2 xâu ký tự có đảm bảo về thứ tự từ điển không?

3) Xâu S gồm n ký tự ($n \leq 255$) có thể tương đương với một mảng một chiều A, khai báo mảng A này như thế nào? Nêu vài điểm khác nhau cơ bản nhất khi sử dụng S và A?

Bài 3.17

Viết chương trình nhập từ bàn phím họ đệm và tên của n học sinh.

1) In lên màn hình danh sách các học sinh được sắp xếp theo alphabe của tên học sinh.

2) In lên màn hình danh sách các học sinh được sắp xếp theo alphabe của tên học sinh, nếu cùng tên thì sắp xếp theo alphabe của họ (của họ đệm).

http://tieulun.hopto.org

Bài 3.18

Viết chương trình nhập từ bàn phím một văn bản gồm một n dòng (n<=10). Cho biết trong văn bản đó:

1) Có bao nhiêu từ?

2) Có bao nhiêu câu?

3) Nguyên âm nào có tần suất xuất hiện cao nhất?

Bài 3.19

Một số có lỗi là một xâu gồm các ký tự là chữ số và ký tự không phải là chữ số. Một số có lỗi được sửa bằng cách xoá đi các ký tự không là chữ số. Nhập từ bàn phím hai số có lỗi, thực hiện việc sửa lỗi và in lên màn hình tổng giá trị của 2 số đó.

IV. DỮ LIỆU KIỂU BẢN GHI

1. Định nghĩa, khai báo bản ghi

1.1. Định nghĩa bản ghi

Bản ghi (record) là một kiểu dữ liệu có cấu trúc bao gồm các thành phần có thể có các kiểu dữ liệu khác nhau gọi là các trường (fields).

Như vậy, bản ghi được sử dụng để biểu diễn các đối tượng có nhiều thuộc tính khác nhau. Ví dụ như đối tượng *học sinh* với các thuộc tính: họ tên, ngày sinh, địa chỉ, điểm thi; đối tượng *hàng hóa* bao gồm các thuộc tính: tên hàng, loại hàng, số lượng, đơn giá...

1.2. Khai báo biến bản ghi

1.2.1. Khai báo gián tiếp

```
Type tên_kiểu_rec = RECORD
  tên_trường_1 : Kiểu_dữ_liệu1;
  tên_trường_2 : Kiểu_dữ_liệu2;
  ...
  tên_trường_n : Kiểu_dữ_liệun;
end;
Var  ten_biến_rec: tên_kiểu_rec;
```

1.2.2. Khai báo trực tiếp

```
Var  tên_biến_rec: RECORD
```


http://tieulun.hopto.org

tên_trường_1 : Kiểu_dữ_liệu$_1$;
tên_trường_2 : Kiểu_dữ_liệu$_2$;
...
tên_trường_n : Kiểu_dữ_liệu$_n$;
end;

1.3. Ví dụ về khai báo biến bản ghi

```
type date = record
   d: 1..31;
   m: 1..12;
   y: integer;
end;
address = record
   n0   : integer;
   street:string[50];
   tell : longint;
end;

HOCSINH = RECORD
   ht: string[30];
   gt: boolean;
   ns: date;
   dc: address;
   dtoan,dly,dhoa: real;
END;
Var
hs1,hs2: HOCSINH;
A: Array[1..100] of HOCSINH;
```

Kiểu dữ liệu của một trường cũng có thể là một kiểu bản ghi đã định nghĩa trước đó.

2. Cách truy nhập các trường của biến bản ghi

Làm việc với biến bản ghi, thực chất là làm việc với các trường.

Một trường được truy nhập (vào/ra, gán) bằng 2 cách sau đây:

http://tieulun.hopto.org

2.1. Cách 1 - Dùng toán tử chấm

Chỉ ra tên biến bản ghi, dấu chấm và tên trường.

Xử lý <Tên_biến_rec.Tên_trường>

Ví dụ 1:

Ta sẽ xem xét các lệnh sau đây để truy nhập đến các ht, gt của biến bản ghi hs1; trường dtoan, dly, dhoa của biến bản ghi A[1]. Lưu ý rằng A là một mảng các phần tử kiểu bản ghi, có nghĩa là bản thân A mới chỉ là một biến mảng, còn A[i] mới thực sự là biến bản ghi.

```
hs1.ht:= 'nguyen minh hang';
hs1.gt:= false;
A[1].dtoan:= 8;
Readln(A[1].dly);
Writeln(A[1].dhoa);
```

Chú ý: Nếu bản thân một trường cũng là một biến bản ghi (con) thì phải tiếp tục truy nhập đến trường của nó sau dấu chấm. Nói cách khác, chỉ có thể trực tiếp làm việc với các trường có kiểu dữ liệu cơ sở như integer, real, char, boolean...

Ví dụ 2:

Trong các lệnh sau đây ta thấy hs1 là một biến bản ghi, hơn nữa hs1.ns cũng vẫn là một biến bản ghi vì ns là một trường kiểu bản ghi . Vậy để truy nhập đến trường d,m,y ta phải viết thông qua biến bản ghi hs1.ns. Tương tự như thế, A[i] và A[i].dc cũng là các biến bản ghi.

```
hs1.ns.d:= 21;
hs1.ns.m:=5;
hs1.ns.y:= 1980;
writeln(A[2].dc.n0);
writeln(A[2].dc.street);
writeln(A[2].dc.tell);
```

2.2. Cách 2 - Dùng câu lệnh with

With Tên_biến_rec do < xử lý Tên_trường>

Các ví dụ 3 và ví dụ 4 dưới đây cho thấy nếu ta muốn truy nhập tới nhiều trường của cùng một biến bản ghi thì nên sử dụng câu lệnh With để tham chiếu trước đến biến bản ghi này.

http://tieulun.hopto.org

Ví dụ 3:

```
with hs2 do
begin
    writeln('ho ten: ',ht);
    if gt then writeln('Nam').else writeln('Nu');
    writeln(dtoan:5:1, dly:5:1,dhoa:5:1);
    with ns do writeln('ngay sinh: ',d, '/',m, '/' ,y);
end;
```

Ví dụ 4:

```
for i:=1 to n do
with A[i] do
begin
   write('ho ten: '); readln(ht);
   write('gioi tinh: '); readln(gt);
   writeln('ngay sinh: '); with ns do readln(d,m,y);
end;
```

3. Các phép toán

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

Hai biến bản ghi cùng kiểu có thể so sánh = và <>

Cũng như các kiểu dữ liệu có cấu trúc, như mảng, không thể vào/ra trực tiếp một biến bản ghi bằng lệnh readln và writeln.

Các phép toán áp dụng cho các trường của một biến bản ghi phụ thuộc vào kiểu dữ liệu của trường đó.

4. Ví dụ bài tập về bản ghi

4.1. Bài toán quản lý tên và lương của các cán bộ

Giả sử mỗi cán bộ gồm:

- Họ đệm
- Tên
- Lương

http://tieulun.hopto.org

Hãy viết chương trình:

1) Nhập từ bàn phím thông tin cho n cán bộ (n<=10).

2) Tìm xem có bao nhiêu người có cùng tên là S. Giá trị của xâu S nhập từ bàn phím. In lên màn hình họ tên đầy đủ của những người đó.

3) Tính và in lên màn hình:

- Lương trung bình của các cán bộ.
- Lương cao nhất.
- Lương thấp nhất.

CANBO.PAS

```
const maxN=10;
Type
   CANBO = record
   hd: string[30]
   ten: string[6];
   luong: real;
   end;
var
   n: integer;
   A: Array[1..maxN] of CANBO;
   i: integer;
   S: string[6];
   d: integer;
   tb,max,min : real;

Begin
   repeat
      write('nhap so can bo: ') readln(n);
until (1<=n) and (n<=maxN);
writeln('nhap thong tin cho cac can bo:');
for i:=1 to n do
with A[i] do
begin
```


http://tieulun.hopto.org

```
        writeln('can bo thu ',i);
        write('ho dem: '); readln(hd);
        write('ten: '); readln(ten);
        write('luong: '); readln(luong);
end;
write('nhap ten can bo S: '); readln(S);
writeln;
writeln('tim cac can bo ten la " ' ,s, ' " ');
d:=0;
tb:= 0;
max:= A[1].luong;
min:= A[1].luong;
for i:=1 to n do
with A[i] do
begin
      if ten=S then
      begin
            inc(d);
        writeln(hd, ' ' ,ten);
        end;
    tb:= tb + luong;
        if max < luong then max:= luong;
        if min > luong then min:= luong;
end;
if d = 0 then writeln('khong co can bo ten la " ',s, ' " ');
writeln('luong trung binh la ',tb/n:0:3);
writeln('luong cao nhat la ' ,max/n:0:3);
writeln('luong thap nhat la ',min/n:0:3);
readln;
end.
```

4.2. Bài toán quản lý tiền điện tiêu dùng

Giả sử mỗi phiếu báo sử dụng điện của một hộ gia đình gồm:

- Tên chủ hộ.

http://tieulun.hopto.org

- Chỉ điện kế tháng trước.
- Chỉ số điện kế tháng này.

Biết rằng đơn giá điện được quy định như sau:

Đơn giá =

{
400 đ/KWh: điện tiêu thụ không quá 100 KW.
500 đ/KWh: điện tiêu thụ trên 100 đến không quá 150 KW.
800 đ/KWh: điện tiêu thụ trên 150 đến không quá 200 KW.
1000 đ/KWh:điện tiêu thụ trên 200 KW.

Viết chương trình:

1. Nhập từ bàn phím n phiếu báo sử dụng điện. (n <= 20).

2. In lên màn hình danh sách gồm tên chủ hộ, lượng điện tiêu thụ và số tiền mà hộ đó phải thanh toán.

3. In lên màn hình danh sách gồm tên các chủ hộ kèm theo lượng điện tiêu thụ theo thứ tự tăng dần của điện tiêu thụ.

TIENDIEN.PAS

```
const
   maxn = 20;
type
   PHIEUDIEN = record
   tenh: string[26];
   cst,csn: integer;
   end;
var
   n: integer;
   A: array[1..maxN] of PHIEUDIEN;
   i,j: integer;
   k,tt: longint;
   tg: PHIEUDIEN;
begin
   repeat
```


http://tieulun.hopto.org

```
        write('nhap so ho: '); readln(n);
until (n>=1) and (n<=maxN);
writeln('nhap n phieu ');
for i:=1 to n do
with A[i] do
begin
      writeln('phieu thu: ',i);
      write('ten chu ho: '); readln(tenh);
      write('chi so dien ke thang truoc: '); readln(cst);
      write('chi so dien ke thang nay: '); readln(csn);
end;
writeln('tien dien cua cac ho gia dinh la: ');
for i:=1 to n do
with A[i] do
begin
    k:= csn - cst;
     if k<=100 then tt:= 400 *k
    else
      if k<=150 then tt:= 100*400 + (k-100)*500
      else
      if k<=200 then tt:= 100*400+50*500+(k-150)*800
      else tt:= 100*400+50*500+ 50*800+(k-200)*1000;
   writeln(ten, ' dien tieu thu ' ,k, ' so tien thanh toan ', tt*1000);
end;

for i:=2 to n do
for j:=n down to i do
if (A[j].csn-A[j].cst) < (A[j-1].csn - A[j-1].cst) then
begin
        tg:= A[j];
        A[j]:=A[j-1];
```


http://tieulun.hopto.org

```
            A[j-1]:=tg;
    end;
    writeln('danh sach cac ho duoc sap xep la:');
    for i:=1 to n do
    with A[i] do writeln(tench,   '   ' ,csn-cst);
    readln;
end.
```

5. Câu hỏi và bài tập

Bài 3.20

Phiếu điểm của một học sinh trong một lớp giả sử gồm:

- Họ tên
- Điểm Môn 1
- Điểm Môn 2
- Điểm Môn 3

Trong đó quy định hệ số các môn: môn 1, môn 2 và môn 3 tương ứng là 1,2,3.

Viết chương trình:

1. Nhập từ bàn phím thông tin cho n phiếu điểm (n <= 20).

2. In lên màn hình danh sách gồm họ tên học sinh, điểm thành phần các môn và điểm trung bình của các môn đó.

3. In lên màn hình danh sách gồm họ tên học sinh cùng với điểm trung bình các môn theo thứ tự tăng dần của điểm trung bình đó.

Bài 3.21

Một hoá đơn cho một khách hàng gồm 1 hoặc nhiều dòng. Mỗi dòng hoá đơn gồm các mục thông tin sau đây:

- Tên hàng
- Số lượng
- Đơn vị
- Đơn giá

Riêng 2 dòng cuối cùng: dòng trên ghi tổng thành tiền, dòng dưới ghi tổng thành tiền sau khi cộng với VAT (10%).

Viết chương trình:

1) Nhập từ bàn phím các thông tin chung của hoá đơn gồm:

http://tieulun.hopto.org

- Số hoá đơn. Họ tên khách. Địa chỉ.

2) Tiếp theo nhập thông tin cho n dòng hoá đơn ứng với n mặt hàng mà khách hàng này mua (n<=10).

3) In lên màn hình tờ hoá đơn đó theo dạng:

HOA DON BAN HANG

So hoa don: 123

Ho ten khach: Nguyen Tran Le

Dia Chi: 113 Hoang Hoa Tham

STT	Ten hang	Don gia	Don vi	So luong	Thanh tien
1	Giay Bai bang	44000	gram	2	88000
2	But Thien long	15000	hop	4	60000
3	Dap ghim	5000	cai	3	15000

Cong tong thanh tien: 163000

Tổng thanh tien sau khi da cong VAT: 179300

Bài 3.22

Viết chương trình:

1) Nhập từ bàn phím sơ yếu lý lịch tóm tắt của N cán bộ. Mỗi một cán bộ gồm các thông tin: Họ tên; Ngày, tháng, năm sinh; Địa chỉ (tên phố, số nhà, điện thoại liên hệ), chức vụ hiện nay.

2) In lên màn hình thông tin vừa nhập.

3) Tìm và in lên màn hình thông tin của một cán bộ sau khi nhập họ tên của cán bộ đó.

4) In lên màn hình danh sách các cán bộ sắp xếp theo thứ tự người già nhất đến người trẻ nhất.

V. DỮ LIỆU KIỂU ĐOẠN CON, LIỆT KÊ VÀ TẬP HỢP

V là phần đọc thêm:

Khi tìm hiểu về mảng, kiểu đoạn con đã được đề cập để mô tả kiểu chỉ số của mảng. Tuy nhiên, chúng ta sẽ tìm hiểu kỹ hơn về kiểu đoạn con cùng với kiểu dữ liệu liệt kê và kiểu tập hợp.

http://tieulun.hopto.org

Thực ra việc giải quyết các bài toán liên quan đến tập hợp đều có thể chuyển về dữ liệu mảng. Tập hợp có nhược điểm là khả năng biểu diễn dữ liệu rất hạn chế (không quá 256 phần tử). Nhưng bên cạnh đó, dữ liệu kiểu tập hợp với các đặc điểm của nó cũng có những ưu điểm nhất định về sự diễn đạt trong sáng, ngắn gọn đối với một số bài toán có một tập không nhiều các phần tử dữ liệu và ta không cần quan tâm đến thứ tự của chúng.

1. Dữ liệu kiểu đoạn con và kiểu liệt kê

1.1. Cách khai báo

1.1.1. Khai báo gián tiếp

```
TYPE
Tên_kiểu_đoạn_con = Giới_hạn_dưới .. Giới_hạn_trên;
Tên_kiểu_liệt_kê  = (giá_trị_1, giá_trị_2,..., giá_trị_n);
VAR
    Tên_biến_kiểu_đoạn_con: Tên_kiểu_đoạn_con;
    Tên_biến_kiểu_liệt_kê: Tên_kiểu_liệt_kê;
```

1.1.2. Khai báo trực tiếp

```
VAR
Tên_biến_đoạn_con: Giới_hạn_dưới .. Giới_hạn_trên;
Tên_biến_liệt_kê  : (giá_trị_1, giá_trị_2,..., giá_trị_n);
```

1.2. Ví dụ về khai báo

```
TYPE
    colors =(red,orange,yellow,green,blue,indigo, magenta);
    ages = 18...59;
VAR
    c1,c2 : colors;
    man1, man2 : Ages;
    job1,job2 : (worker, engineer, doctor, teacher);
    w1,w2: 30..150;
```

1.3. Cách sử dụng

Dữ liệu kiểu liệt kê không vào/ra trực tiếp bằng lệnh ReadIn và WriteIn;

Các kiểu dữ liệu đoạn con và liệt kê có thể thực hiện lệnh gán và so sánh.

http://tieulun.hopto.org

Nói riêng hàm ord cho biết thứ tự của các giá trị trong kiểu liệt kê bắt đầu được tính từ vị trí 0. Các thủ tục Pred, Succ được áp dụng trên kiểu dữ liệu có thứ tự như kiểu liệt kê.Chương trình sau đây được chấp nhận:

```
Var i: (Lan,Linh,Lien,Le,Lam);
Begin
  For i:=Lan To Lam Do Write(ord(i)+1,'  '); Readln;
End.
```

Các kiểu dữ liệu liệt kê và đoạn con thường được sử dụng để tham gia định nghĩa kiểu dữ liệu có cấu trúc do người lập trình định nghĩa để xử lý trên các kiểu dữ liệu có cấu trúc mạnh hơn này.

2. Dữ liệu kiểu tập hợp

2.1. Cách khai báo tập hợp

2.1.1. Khai báo trực tiếp

VAR Tên_biến_tập_hợp: SET OF Kiểu_cơ_sở;

2.1.2. Khai báo gián tiếp

TYPE Tên_kiểu_tập_hợp = SET OF Kiểu_cơ_sở;

VAR Tên_biến_tập_hợp: Tên_kiểu_tập_hợp;

Chú ý:

Kiểu_cơ_sở (kiểu dữ liệu của tập hợp) tức là một kiểu dữ liệu chuẩn hoặc kiểu dữ liệu đơn giản tự định nghĩa như kiểu đoạn con, kiểu liệt kê. Tuy nhiên, kiểu dữ liệu được chọn có phạm vi không vượt quá 256 giá trị phân biệt.

Vậy có các kiểu dữ liệu sau đây có thể mô tả cho kiểu dữ liệu tập hợp:

- Byte: 0..255.
- Char: #0..#255.
- Boolean: FALSE..TRUE.
- Các kiểu đoạn con, khoảng con với không quá 256 giá trị phân biệt.

2.2. Ví dụ khai báo tập hợp

```
TYPE
    Colors = (red, orange, yellow, green, blue, indigo, magenta);
    Codes = 1...1000;
    ColorSet = set of colors;
VAR
```


http://tieulun.hopto.org

```
C1, C2, C3 : ColorSet;
idx1, idx2: set of Codes;
A,B,C : set of byte;
M,N,P: set of char;
```

2.3. Các phép toán trên tập hợp

2.3.1. Phép gán

```
C1:= [red,blue, yellow];{gán theo tập giá trị}
C2:= [ ]; {khởi tạo tập rỗng}
C3:= C1; {gán 2 biến tập hợp}
A:  [1..10,15..30,43,47];
```

2.3.2. Phép so sánh

Hai tập hợp bằng nhau nếu chúng có các phần tử giống nhau và không phân biệt về thứ tự của các phần tử trong một tập hợp.

```
M := [ 'A', 'D', 'P', 'O'];
N:= [ 'D', 'O', 'A', 'P'];
```

Suy ra M = N

2.3.3. Phép hợp

```
A := [1,3,7,10];
B :=[5,7,4,9,3];
C := A+B;
```

C = [1,3,4,5,7,9,10];

Nói riêng

A:= A + [19] ; Suy ra A= [1,3,7,10,19];

Nhưng nếu x:= 19; thì ta có thể viết

A:= A + [x];

2.3.4. Phép giao

C:= A*B; Suy ra C = [3,7];

2.3.5. Phép lấy phần bù

C:= A-B; Suy ra C = [1,10];

C:=B-A; Suy ra C = [5,4,9];

http://tieulun.hopto.org

2.3.6. *Phép thuộc (in)*

Nếu có x: byte, ta có thể viết:

If x in A then Lệnh;

If not (x in A) then Lệnh;

2.4. Một ví dụ về tập hợp

Một lớp có không quá 26 học sinh, ký hiệu các học sinh bằng các chữ cái in hoa và bắt đầu từ 'A' trở đi.

1. Nhập vào sĩ số lớp và số hiệu các học sinh tham gia tiết mục hát, số hiệu các học sinh tham gia tiết mục múa.

2. In lên màn hình các học sinh tham gia cả 2 tiết mục múa và hát.

3. In lên màn hình danh sách các học sinh không tham gia tiết mục nào.

VANNGHE.PAS

```
Const MaxN=26;
Type
   List = 'A'..'B';
   Group = Set of List;
Var  n,i: byte;
     hsd,hsc, k: char;
     LOP, M, H, HM: Group;
Begin
   Repeat
        Write ('Nhap si so lop: ');
    Repeat
        Readln(n);
         if not (n in [1..MaxN]) then write('Nhap lai n: ');
   Until (n in [1..MaxN]);
   hsd:= 'A'; hsc:= chr(64+n);
   LOP:= [hsd..hsc];
   M:=[];
      Writeln('Nhap cac hoc sinh tham gia hat:'); i:=1;
   Repeat
```


http://tieulun.hopto.org

```
                Write('Nhap hoc sinh thu:',i,'nhap 0 de dung lai ');
                Readln(k);
                if (k in LOP) then
                begin
                M:= M + [k]; inc(i);
                end;
    Until k='0';
    H:=[];
    Writeln('Nhap cac hoc sinh tham gia mua:');
    i:=1;
    Repeat
                Write('Nhap hoc sinh thu: ',i,'nhap 0 de dung lai ');
                Readln(k);
                if (k in LOP) then
                begin
                H:= H + [k]; inc(i);
                end;
    Until k='0';
    Writeln('Danh sach cac hs tham gia ca 2 tiet muc la: ');
    for k:= hsd to hsc do
    if (k in M*H) then Write(k, ','); Writeln;
    Writeln('Danh sach cac hs khong tham van nghe la: ');
    for k:= hsd to hsc do
    if (k in LOP-M-H) then Write(k, ','); Readln;
End.
```

3. Câu lệnh chọn lựa

3.1. Cú pháp và ý nghĩa của câu lệnh chọn lựa case

Câu lệnh chọn lựa case dùng để thay cấu trúc rẽ nhánh khi phải biện luận nhiều trường hợp rời rạc.

Cú pháp của lệnh case như sau:

Case biến of

giá_trị_1: begin nhóm_lệnh_1; end;

http://tieulun.hopto.org

giá_trị_2: begin nhóm_lệnh_2; end;
...
giá_trị_n: begin nhóm_lệnh_n; end;
else begin nhóm_lệnh_khác; end;
End;

Cấu trúc case nói trên thay thế lệnh rẽ nhánh sau đây:

If biến=giá_trị_1 then begin nhóm_lệnh_1; end
Else
If biến=giá_trị_2 then begin nhóm_lệnh_2; end
else
...
else
If biến=giá_trị_n then begin nhóm_lệnh_n; end
else begin nhóm_lệnh_khác;
end;

3.2. Một số ví dụ về lệnh case

3.2.1. Bài toán tính lương theo ngày công

Viết chương trình tính lương cho một nhân viên theo ngày công và mức lương trong một ngày được quy định theo chức vụ (ký hiệu là một ký tự) như sau: Lương/1 ngày công =

{
50.000 nếu là giám đốc (ký hiệu là G).
40.000 nếu là phó giám đốc (ký hiệu là P).
30.000 nếu là trưởng phòng (ký hiệu là T).
25.000 nếu là phó phòng (ký hiệu là O).
20.000 nếu là nhân viên (ký hiệu là N).
10.000 nếu là bảo vệ hoặc phục vụ (ký hiệu là B).

```
Var    songay   : byte;
         cv     : char;
       chucvu   : set of char;
      lgngay,lgthang : longint;
Begin
  Write('Nhap so ngay cong: ') ;
```


http://tieulun.hopto.org

```
    Repeat
     Readln(songay);
     If not (songay in [ 1..30]) then
                write('Nhap lai: ');
    Until (songay in [1..30]);
    chucvu:= ['G', 'P', 'T', 'O', 'N', 'B' ] ;
    Write('Nhap chuc vu: ') ;
     Repeat
      Readln(cv);
     If not (cv in chucvu) then
              Write('Nhap lai: ');
    Until cv in chucvu;
    Case cv of
    'G': lgngay:= 50000;
    'P': lgngay:= 40000;
    'T': lgngay:= 30000;
    'O': lgngay:= 25000;
    'N': lgngay:= 20000;
    'B': lgngay:= 10000;
    End;
    Lgthang:= lgngay * songay;
    Writeln('Luong cua nhan vien do la: ', lgthang);
    Readln;
 End.
```

Chú ý:

Chỉ áp dụng câu lệnh chọn lựa case trong trường hợp phải biện luận khá nhiều khả năng rời rạc nhau. Trong những trường hợp các khả năng được biểu thị bởi khoảng của dãy giá trị không đếm được (ví dụ như giá trị kiểu real) thì vẫn nên dùng câu lệnh if để tiện biện luận. Chẳng hạn, bài toán sau đây phải dùng cấu trúc if:

Nhập từ bàn phím điểm trung bình của một học sinh và in lên màn hình xếp loại học lực của học sinh đó theo quy định như sau:

http://tieulun.hopto.org

Xếp loại là :

{ Giỏi nếu TBM >= 8.0
Khá nếu 6.5<= TBM <8.0
TB nếu 5.0<= TBM < 6.5
Yếu nếu TBM <5.0

3.2.2. Bài toán minh hoạ cách tạo menu đơn giản trong chương trình

Viết chương trình nhập vào 3 số thực a,b,c sau đó tạo một menu trong đó cho phép người dùng chọn lựa một trong các yêu cầu sau đây:

1. Tính và in lên màn hình chu vi, diện tích tam giác 3 cạnh a,b,c.
2. Giải phương trình ax + b = c.
3. Tính và in lên màn hình giá trị đa thức:

$ax^3 + b x^2 + cx + abc$

với các giá trị khác nhau của x nhập từ bàn phím.

Chương trình có khả năng chạy được liên tục.

uses CRT;

```
uses CRT;
Var   chon   : byte;
      a,b,c  : real;
      p,q, x : real;
Begin
 REPEAT
   Clrscr;
   Write('Nhap 3 so thuc a,b,c: ');
   Readln(a,b,c);
   Writeln;
   Writeln('1. Tinh chu vi dien tich tam giac abc');
   Writeln('2. Giai phuong trinh ax + b = c');
   Writeln('3. Tinh f(x) = ax^3 + bx^2 + cx + abc ');
   Writeln('4. Thoat khoi chuong trinh ');
   Repeat
     Write('Ban chon muc nao: '); Readln(chon);
   Until (chon in [1..4]);
```


http://tieulun.hopto.org

```
Case chon of
1: begin
 if (a<0) or (b<0) or (c<0) or
   (a+b <=c) or (a+c<=b) or (b+c <=a)
 then writeln('a,b,c khong tao thanh tam giac ')
 else
 begin p:= (a+b+c)/2;
      q:= sqrt(p*(p-a)*(p-b)*(p-c));
      writeln('chu vi tam giac la: ', 2*p: 0: 3);
      writeln('dien tich tam giac la: ', q:0:3);
  end;
 2: { ax = c-b }
   if a=0 then
    if c-b=0 then write('phuong trinh vo so nghiem')
    else writeln('phuong trinh vo nghiem')
   else writeln('phuong trinh co nghiem x = ',
(c-b)/a:0:3);
 3: { ax^3 + bx^2 + cx + abc }
   begin
    write('Nhap x = '); Readln(x);
    q:= a*x*x*x + b*x*x + c*x + a*b*c ;
    writeln('gia tri f(' ,x:0:3,  ') = ' ,q:0:3);
   end;
 4: writeln('Godd bye!');
End;
Readln;
Until chon=4;
End.
```

4. Câu hỏi và bài tập

Bài 3.23

- Có thể nhập giá trị cho dữ liệu kiểu liệt kê bằng lệnh readln được không?

- Kiểu đoạn con trong Pascal có thể biểu diễn một trục số có bao nhiêu điểm nguyên?

http://tieulun.hopto.org

- Liệu có thể khai báo một biến tập hợp kiểu có kiểu dữ liệu của tập hợp là kiểu số nguyên không?

Bài 3.24

Nếu A là một biến tập hợp kiểu byte, xét tính đúng sai của các lệnh sau đây:

A:=0;

A:='';

A:=[];

A:=5;

A:=[5];

A:= A + [5..9];

Bài 3.25

Có n đề tài ký hiệu là các số tự nhiên, n<=200. Có m sinh viên tham gia viết đề tài. Một đề tài có thể có nhiều sinh viên. Một sinh viên, về nguyên tắc có thể viết nhiều đề tài, nhưng không được quá 5 đề tài.

Viết chương trình:

1. Nhập n, m và các đề tài cho từng sinh viên.

2. In lên màn hình số hiệu các đề tài mà tất cả các sinh viên đều tham gia viết.

3. In lên màn hình số hiệu các đề tài mà tất cả các sinh viên đều né tránh.

Bài 3.26*

Một lớp có n học sinh, tên các học sinh ký hiệu là các chữ cái in hoa (n<=26) bắt đầu từ 'A' đến CHR(64+n).

Biết trước m quan hệ có dạng sau:

AB → CDE

Có nghĩa là nhóm các bạn {A, B} thì có thể thuyết phục được các bạn {C,D,E} đi tham quan.

Viết chương trình:

1. Nhập từ bàn phím số n là sĩ số lớp và m quan hệ dạng trên. Với mỗi quan hệ, tổ chức nhập tên các học sinh vế trái và tên các học sinh vế phải của quan hệ đó.

2. Với X là nhóm các học sinh nhập từ bàn phím, hãy cho biết nhóm X sẽ rủ được những ai đi tham quan.

3. Tìm một nhóm học sinh nòng cốt của lớp, đó là nhóm ít học sinh nhất nhưng có khả năng rủ được cả lớp đi chơi.

http://tieulun.hopto.org

Chương 4

CHƯƠNG TRÌNH CON

I. CHƯƠNG TRÌNH CÓ CHƯƠNG TRÌNH CON

1. Khái niệm chương trình con

1.1. Khái niệm chương trình con

Chương trình con về mặt bản chất cũng như một chương trình bình thường. Về mặt hình thức, ngoài phần khai báo và phần thân chương trình thì chương trình con phải có tên gọi (không như chương trình chính, có thể bỏ qua phần tiêu đề là tên gọi của nó). Ngoài ra, khi thực hiện, chương trình con không trực tiếp chạy được (bằng lệnh run hay Control - F9) mà chỉ thực hiện khi có một chương khác gọi nó thông qua tên của nó.

1.2. Phân loại chương trình con

Có 2 loại chương trình con: thủ tục (procedure) và hàm (function).

Cách xây dựng một thủ tục như sau:

```
Procedure Tên_thủ_tục (khai_báo_các_tham_số) ;
khai_báo_địa_phương;
begin
Các_lệnh_của_thủ_tục;
end;
```

Cách xây dựng một hàm như sau:

```
Function Tên_hàm (khai_báo_các_tham_số): Kiểu_dữ_liệu_hàm;
Khai_báo_địa_phương;
begin
Các_lệnh_của_hàm;
Tên_hàm:= biểu_thức_cần_tính;
end;
```


http://tieulun.hopto.org

Chú ý: Chương trình con (Thủ tục và hàm) có thể không có tham số.

1.3. Ý nghĩa của chương trình con

Một trong các phương pháp tiếp cận trong lập trình là "lập trình đi xuống" (phương pháp làm mịn dần). Trong phương pháp tiếp cận này, bài toán cần giải quyết được phân rã thành một số bài toán con. Mỗi bài con lại có thể tiếp tục chia thành các bài toán con nhỏ hơn nữa. Quá trình phân rã như vậy dừng lại đến khi thu được các bài toán con "mịn nhất", nghĩa là nó giải quyết một cách dễ dàng bằng các thuật toán đã biết. Mỗi một bài toán con có thể được xây dựng bằng một chương trình con.

Có thể có những chương trình con hoàn toàn độc lập với chương trình chính, nghĩa là nó được xây dựng không phụ thuộc vào bất kỳ chương trình nào và có thể được vận dụng nhiều lần trong một chương trình nào đó hoặc trong nhiều chương trình khác nhau.

Các ví dụ dưới đây tạm thời chưa quan tâm chi tiết đến cách khai báo các tham số của chương trình con.

Ví dụ 4.1

Xây dựng hàm UCLN nhận vào 2 số nguyên a và b và cho kết quả là USCLN của a và b:

```
function USCLN(a,b: integer): integer;
var r: integer;
begin
r:= a mod b;
while r>0 do
begin
a:=b;
b:=r;
r:=a mod b;
end;
USCLN:= b;
end.
```

Rõ ràng trong bất kỳ chương trình nào cần tìm ước số chung lớn nhất của 2 số x,y nào đó ta có thể gọi thực hiện hàm nói trên. Chẳng hạn để in lên màn hình USCLN của 2 số x,y đó, ta viết lệnh:

http://tieulun.hopto.org

```
write(USCLN(x,y));
```

Ví dụ 4.2

Xây dựng thủ tục nhập từ bàn phím một số nguyên a thỏa mãn $d \le a \le c$.

```
Procedure nhapso(var a: integer; d,c: integer);
Begin
Write('nhap so: ');
repeat
readln(a);
if (a<d) or (a>c) then write('nhap lai: ');
until (a>= d) and (a<=c);
End;
```

Thủ tục này sẽ được sử dụng khá nhiều khi trong chương trình ta cần nhập dữ liệu có điều kiện. Chẳng hạn, để nhập n trong đoạn [1..100] ta sẽ viết lệnh:

```
Nhapso(n,1,100);
```

Chương trình con được ví như một hộp đen gồm đầu vào và đầu ra. Muốn sử dụng tốt chương trình con thì phải biết cách nhìn nó ở 2 góc độ:

- Xây dựng chương trình con được ví như trả lời câu hỏi là bên trong cái hộp đen đó cái cái gì, thiết kế như thế nào? Tại sao ?
- Gọi thực hiện chương trình con được ví như trả lời câu hỏi: cái hộp đen này dùng như thế nào? Tác dụng gì?

Rõ ràng 2 câu hỏi "tại sao?" và "tác dụng gì?" khác hẳn nhau. Nghĩa là khi một chương trình gọi sử dụng một chương trình con A thì nó chỉ quan tâm tới tác dụng của chương trình con A, biết cách sử dụng A mà không cần phải quan tâm xem A được làm như thế nào, tại sao lại như thế. Một số học sinh khó khăn trong việc sử dụng chương trình con vì đến khi gọi thực hiện chương trình con lại cứ "băn khoăn" tìm cách trả lời câu hỏi "tại sao?". Câu hỏi này đã được trả lời trong lúc xây dựng chương trình con rồi.

2. Cấu trúc một chương trình có chương trình con

Chương trình có tổ chức chương trình con gồm ba phần viết theo thứ tự sau đây:

http://tieulun.hopto.org

- Khai báo toàn cục.
- Xây dựng các chương trình con (hàm và thủ tục).
- Thân chương trình chính.

```
<Các_khai_báo_toàn_cục>;
Procedure Tên_thủ_tục_1 (khai_báo_các_tham_số) ;
khai_báo_địa_phương;
begin Các_lệnh_của_thủ_tục_1;
end;

Procedure Tên_thủ_tục_2 (khai_báo_các_tham_số) ;
khai_báo_địa_phương;
begin Các_lệnh_của_thủ_tục_2;
end;
...
Function Tên_hàm_1(Các_tham_số): Kiểu_dữ_liệu_hàm1;
khai_báo_địa_phương;
begin Các_lệnh_của_hàm;
end;
...
BEGIN
Các_lệnh_của_chương_trình_chính;
< trong đó có các lời gọi thực hiện các thủ tục và hàm>
END.
```

3. So sánh thủ tục và hàm

Trong ngôn ngữ Pascal, khái niệm thủ tục là tường minh và không thể nhầm lẫn với hàm. Mặc dù với Turbo Pascal 7.x có thể sử dụng hàm như một mệnh lệnh (theo kiểu gọi thực hiện một phương thức như hướng đối tượng với C++), chúng ta không khuyến khích sử dụng hàm theo cách này đối với lập trình Pascal.

Thực ra, chúng ta đã từng tiếp cận với thủ tục và hàm có sẵn trong ngôn ngữ Pascal. Ví dụ abs, sqrt, length, chr... là các hàm, còn readln, writeln, inc, dec, delete, insert,... là các thủ tục. Đối với các chương trình con có sẵn, này chúng ta chỉ cần quan tâm đến tác dụng của nó cũng như cách sử dụng.

http://tieulun.hopto.org

Thủ tục và hàm giống nhau ở chỗ, chúng đều là chương trình con có vai trò như những modul chương trình. Thủ tục và hàm về cơ bản khác nhau ở những điểm sau đây:

	Thủ tục	Hàm

Cách sử dụng	Thú tục dùng như mệnh lệnh.
	Ví dụ:
	Writeln(' Comparison !');
	Val(n,s,c);
	Nhapdl;
	Inkq;
		Hàm dùng như một biểu thức.
		Ví dụ:
		a:= sqrt(x*x + y*y +1);
		n:= Length(s);
		s:= dientich(a,b,c);

Đặc điểm	Tên thủ tục không có giá trị do đó thủ tục không có khai báo kiểu dữ liệu như hàm.	- Tên của hàm có giá trị nên nó phải được khai báo kiểu dữ liệu (không cấu trúc). - Hàm luôn có lệnh gán để trả lại giá trị cần tính cho tên hàm.

Chú ý: Lệnh gán tên hàm bằng biểu thức cần tính chỉ được thực hiện một lần khi hàm được triệu gọi, mặc dù lệnh gán đó trong thân hàm có thể viết nhiều lần (chẳng hạn trong cấu trúc rẽ nhánh đủ).

Ví dụ 4.3

Viết chương trình nhập 3 số nguyên a, b, c. Tính chu vi và diện tích tam giác đó.

```
Var a,b,c: integer;
Procedure nhapso(var a: integer; d,c: integer);
Begin
repeat
readln(a);
if (a<d) or (a>c) then write('nhap lai: ');
until (a>= d) and (a<=c);
End;
```


http://tieulun.hopto.org

```
Function cv(a,b,c: integer): real;
Begin
  cv:= a+b+c;
End;
Function dt(a,b,c: integer): real;
var p: real;
begin
  p:= cv(a,b,c)/2;
  dt:= sqrt(p* (p-a)* (p-b)* (p-c);
end;
Begin
  Write('Nhap canh a: '); Nhapso(a,1,maxint);
  Write('Nhap canh b: '); Nhapso(b,1,maxint);
  Write('Nhap canh c: '); Nhapso(c,1,maxint);
  if (a+b<=c) or (a+c<=b) or (b+c<=a) then
   writeln('a,b,c khong tao thanh tam giac' )
 else
 begin
  writeln('chu vi tam giac la: ', cv(a,b,c): 0: 3);
  writeln('dien tich tam giac la: ', dt(a,b,c): 0: 3);
 end;
  readln;
End.
```

4. Câu hỏi và bài tập

Bài 4.1

- Tác dụng của việc tổ chức chương trình thành chương trình con?

- Phân biệt sự khác nhau giữa thủ tục và hàm?

- Trong thân chương trình của một hàm, lệnh gán tên hàm bằng các giá trị trong quá trình tính toán có thể được thực hiện nhiều lần không?

Bài 4.2

Viết chương trình trong đó có xây dựng và sử dụng hàm tìm ước số chung lớn nhất của 2 số nguyên dương để:

http://tieulun.hopto.org

1) Nhập một mảng gồm n số nguyên dương (n<=20).

2) In lên màn hình ước số chung của các số trong mảng.

Bài 4.3

Viết chương trình: Nhập từ bàn phím toạ độ 4 đỉnh của một tứ giác lồi sau đó in lên màn hình diện tích của tứ giác này.

Trong chương trình cần xây dựng và sử dụng các thủ tục và hàm sau đây:

1. Thủ tục Nhapso(a); nhập một số dương a.

2. Hàm dodai(M,N); tính độ dài đoạn thẳng đi qua 2 điểm M,N. M, N là 2 biến bản ghi gồm 2 thành phần x,y biểu thị toạ độ của một điểm.

3. Hàm dientichtg(a,b,c); tính diện tích của một tam giác biết độ dài 3 cạnh là a,b,c.

II. THAM BIẾN VÀ THAM TRỊ, BIẾN TOÀN CỤC VÀ BIẾN ĐỊA PHƯƠNG

1. Biến toàn cục và biến địa phương: So sánh, cách sử dụng

Về hình thức: Biến toàn cục khai báo ở chương trình chính nghĩa là không khai báo trong bất kỳ chương trình con nào. Ngược lại, biến địa phương là các biến khai báo trong các chương trình con.

Về phạm vi sử dụng: Nói chung, biến toàn cục được khai báo ngay từ đầu chương trình chính và do đó nó có tác dụng trong toàn bộ chương trình. Một cách tổng quát, khi biến toàn cục được khai báo ở một vị trí nào đó thì nó chỉ không có tác dụng đối với những chương trình con được khai báo trước vị trí của nó.

Biến địa phương chỉ có tác dụng trong phạm vi chương trình con nơi nó được khai báo, nghĩa là trong thân chương trình con đó và các chương trình "con, cháu" của nó.

Khi tên của biến địa phương trùng với tên biến toàn cục thì biến toàn cục bị che mất, nghĩa là chương trình con chứa biến này sẽ không sử dụng được những giá trị của biến đó với tư cách nó là biến toàn cục.

Ví dụ 4.4

Tìm chỗ sai trong chương trình sau đây:

http://tieulun.hopto.org

```
var
a: integer;
procedure ABC;
var b: integer;
begin
  b:=2*a;
end;
begin
  a:=10;
  ABC;
  writeln(a);
  writeln(b);
  readln;
end.
```

Lệnh writeln(b) sai vì biến địa phương b chỉ được phép sử dụng trong chương trình con ABC;

Biến a là toàn cục nên sau khi nó nhận giá trị 10, chương trình con ABC sẽ được phép sử dụng nó và tính được b=20.

Ví dụ 4.5

Tìm chỗ sai trong chương trình sau đây:

```
Procedure ABC;
var b: integer;
begin
  b:=2*a;
end;
var a: integer;
begin
  a:=10;
  ABC;
  writeln(a);
readln;
end.
```

Việc sử dụng biến a trong chương trình con ABC là sai vì a khai báo sau khai báo chương trình con ABC.

http://tieulun.hopto.org

Ví dụ 4.6

Giải thích hoạt động của chương trình sau đây:

```
var a,b: integer;
procedure ABC;
var a,n: integer;
   procedure XYZ;
   var p: integer;
   begin
        a:= a+1;
        b:= b+1;
        p:= a*a;
        n:= p + b;
   end;
begin
a:= 5;
XYZ;
writeln(n);
end;
Begin
 a:=10; b:=20;
 ABC;
 writeln(a);
 writeln(b);
End.
```

Nói chung khi giải thích hoạt động của một chương trình, ta cũng đi theo thứ tự như lúc phân rã chương trình thành các chức năng nhỏ hơn. Nghĩa là ta bắt đầu giải thích từ chương trình chính:

Ban đầu a và b khởi gán bằng 10 và 20.

Thủ tục ABC được triệu gọi. Biến a và n là các biến địa phương. Biến địa phương a sẽ che mất biến toàn cục a, vì vậy ABC sẽ không sử dụng được giá trị a = 10 ở chương trình chính. Xét thân thủ tục ABC, ban đầu khởi tạo a = 5 và thủ tục XYZ được gọi thực hiện.

http://tieulun.hopto.org

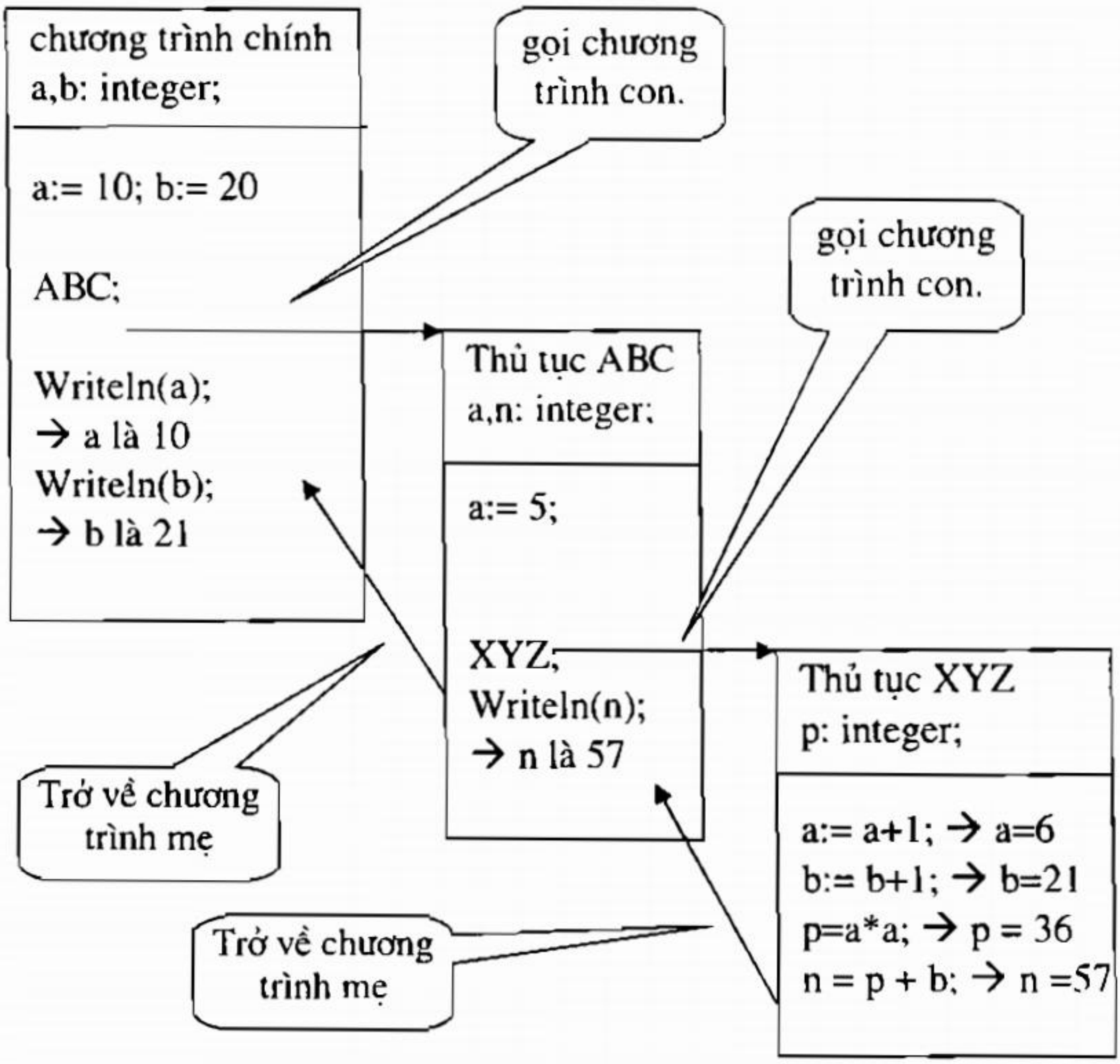
Thủ tục XYZ nhận biến b là toàn cục, vậy nó hiểu b = 20. Mặt khác nó cũng được kế thừa từ chương trình mẹ hai biến a và n, với giá trị của a = 5.

Sau khi tăng a = 6, b = 21 thì $p = a^2 = 36$, n = p + b = 36 + 21 = 57.

Thủ tục XYZ chấm dứt và quyền điều khiển trả về lệnh sau lệnh gọi thực hiện XYZ trong thân chương trình của ABC. Nghĩa là lệnh writeln(n) sẽ in lên màn hình số 57.

Thủ tục ABC chấm dứt, quyền điều khiển trả về chương trình chính. Hai lệnh writeln(a) và writeln(b) thực hiện sẽ in lên màn hình giá trị của biến toàn cục a = 10 (các chương trình con ABC và XYZ không sử dụng giá trị này của a, vì nó bị biến địa phương trùng tên che mất) và b = 21 (đã bị chương trình cháu XYZ thay đổi).

Hoạt động của chương trình trên minh hoạ như hình sau đây:



http://tieulun.hopto.org

2. Tham biến và tham trị: So sánh, cách sử dụng

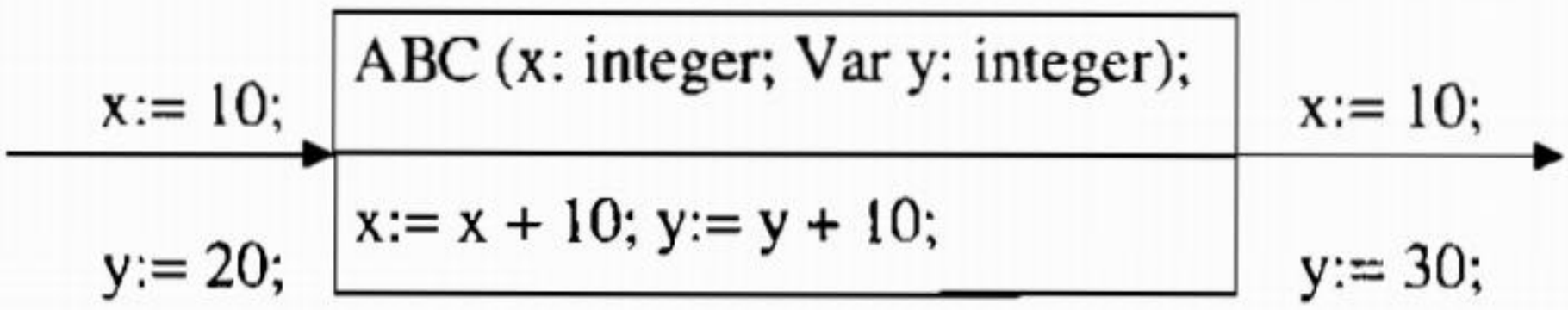
Phần khai báo ở trong ngoặc đơn ngay sau tên chương trình con (không phải là biến địa phương của chương trình con đó) được gọi là khai báo tham số.

Khi xây dựng chương trình con, các tham số đó gọi là các **tham số hình thức**. Khi chương trình con được gọi sử dụng, các giá trị hay các biến "có thật" của chương trình "mẹ" sẽ thay thế các tham số hình thức và khi đó ta gọi chúng là **tham số thực sự**.

Tham số (hình thức hay thật sự) có hai loại: Tham số biến và tham số giá trị, gọi tắt là **tham biến** và **tham trị**.

Về mặt hình thức: Tham số nào được khai báo sau từ khóa Var gọi là tham biến, ngược lại, tham số nào không được khai báo sau từ khóa var gọi là tham trị.

Về mặt hoạt động: Tham trị thì giữ nguyên giá trị đầu vào khi ra khỏi chương trình con, mặc dù nó bị thay đổi giá trị trong thân chương trình con đó. Ngược lại, tham biến thì giữ lại giá trị đã bị chương trình con làm thay đổi khi ra khỏi chương trình con. Nguyên tắc này được tóm tắt qua ví dụ ở hình sau đây:



Ví dụ 4.7

Thủ tục:

```
Procedure nhapso(var a: integer; d,c: integer);
Begin
repeat
readln(a);
if (a<d) or (a>c) then write('nhap lai: ');
until (a>= d) and (a<=c);
End;
```


http://tieulun.hopto.org

Phải khai báo a là tham biến. Vì, nếu a là tham trị, trước khi gọi thủ tục nhapso(a). giá trị của a là không xác định. Trong thủ tục, cho dù ta có nhập a bằng bao nhiêu đi nữa thì khi ra khỏi thủ tục, giá trị của a vẫn giữ nguyên như đầu vào, nghĩa là có giá trị không xác định và việc nhập a trong thủ tục là vô ích.

Ví dụ 4.8

Viết chương trình tìm USCLN của dãy n số nguyên dương nhập từ bàn phím, (N <= 20).

Thủ tục procedure USCLN(a: integer; var b: integer) có tác dụng tìm USCLN của a và b và gán kết quả tìm được cho b. Vậy b phải là tham biến để chương trình chính sử dụng nó.

```
Var  i, n: integer;
     a: array[1..20] of integer;
Procedure nhapso(var a: integer; d,c: integer);
Begin
repeat
readln(a);
     if (a<d) or (a>c) then write('nhap lai: ');
until (a>= d) and (a<=c);
End;

procedure USCLN(a: integer; var b: integer);
var r: integer;
begin
     r:= a mod b;
   while r<>0 do
 begin
              a:= b; b:= r; r:= a mod b;
 end;
end;
Begin
 write('Nhap n: '); Nhapso(n,1,20);
```


http://tieulun.hopto.org

```
  for i:=1 to n do
  begin
   write('a[' ,i, ']= '); nhapso(a[i],1,maxint);
  end;
  for i:=2 to n do USCLN(a[i], a[1]);
  writeln('USCLN = ', a[1]); Readln;
End.
```

Lưu ý: Ta có thể dùng hàm thay vì dùng thủ tục như sau:

```
function USCLN(a,b: integer): integer ;
var r: integer;
begin
 r:= a mod b;
 while r<>0 do
 begin
    a:= b;
    b:= r;
    r:= a mod b;
  end;
 USCLN:= b;

end;
```

Và khi đó, ở chương trình chính, khai báo thêm u: integer là USCLN cần tìm, ta có:

```
  u:= a[1];
  for i:= 2 to n do u:= USCLN(u,a[i]);
```

Nhận xét:

Khi nào dùng tham biến và khi nào dùng tham trị ? Câu trả lời nằm ngay ở sự so sánh về hoạt động của tham biến và tham trị. Nghĩa là khi ta không có ý định thay đổi giá trị của tham số trong thân hàm hay thủ tục thì tham số đó là tham trị. Ngược lại nếu ta có ý định gọi một hàm hay thủ tục với ý đồ tính toán và giữ lại kết quả của tham số truyền vào thì tham số đó khai báo là tham biến.

http://tieulun.hopto.org

Nếu chương trình con trả lại một giá trị thuộc kiểu đơn giản chuẩn thì nên sử dụng hàm. Nếu ngược lại thì nên dùng thủ tục, nghĩa là khi giá trị trả lại của chương trình con thuộc một kiểu dữ liệu có cấu trúc hoặc trả lại nhiều giá trị khác nhau thì ta sẽ khai báo tương ứng với các tham số biến trong một thủ tục.

3. Tham chiếu (đọc thêm)

Trong ngôn ngữ Pascal còn cho phép truyền tham số kiểu tham chiếu, nghĩa là truyền vào chương trình con địa chỉ của biến. Thường thì địa chỉ này được xác định bởi con trỏ cùng kiểu dữ liệu với tham số thực sự. Như vậy về mặt tác dụng thì kiểu truyền theo tham chiếu giống như kiểu truyền theo tham số biến.

Ví dụ 4.9

```
Type
IntPoint = ^ integer;
Var
x: integer;
p: IntPoint;
Procedure Test(p: IntPoint);
Begin
 p^:= p^+20;
End;
Begin
 x:=10;
 Test(@x); {@ là phép toán lấy địa chỉ của biến x}
 writeln(x); {Kết quả x = 30}
End.
```

4. Câu hỏi và bài tập

Bài 4.4

Viết chương trình:

- Nhập từ bàn phím 2 số a, b (1<= a <= b <= 500).
- In lên màn hình tất cả các số nguyên tố trong đoạn [a,b].

http://tieulun.hopto.org

Trong chương trình cần xây dựng một hàm nhận vào một số nguyên x và hàm cho giá trị True nếu x là số nguyên tố, cho giá trị False nếu x là hợp số.

Bài 4.5

Viết chương trình trong đó có xây dựng một hàm tính đơn giá điện tiêu thụ, biết chỉ số điện tiêu thụ của tháng hiện tại và tháng trước đó theo quy định ở bài tập 2.12. Vận dụng hàm này để thông báo tiền điện phải trả cho n hộ gia đình. Thông tin về các hộ được nhập từ bàn phím.

Bài 4.6

Lương cho một nhân viên theo ngày công và mức lương trong một ngày được quy định theo chức vụ (ký hiệu là một ký tự) như sau:

Lương/1 ngày công =

{
50.000 nếu là giám đốc (ký hiệu là G).
40.000 nếu là phó giám đốc (ký hiệu là P).
30.000 nếu là trưởng phòng (ký hiệu là T).
25.000 nếu là phó phòng (ký hiệu là O).
20.000 nếu là nhân viên (ký hiệu là N).
10.000 nếu là bảo vệ hoặc phục vụ (ký hiệu là B).
}

Hãy viết chương trình tính lương trong một tháng cho n nhân viên (1<= n<=39) biết số ngày công và chức vụ của từng người nhập từ bàn phím. Ngoài ra trong lương tổng cộng còn bao gồm cả tiền thưởng theo quy định như sau:

Thưởng =

{
200 nếu số ngày công >= 26.
100 nếu 20<= số ngàycông <26.
50 nếu 15<= số ngàycông <20.
}

Yêu cầu về tổ chức chương trình: Trong chương trình phải có các thủ tục và hàm sau đây:

- Thủ tục nhập một số nguyên x thỏa mãn l<= x <= r. l và r được xem như là các giá trị biết trước truyền vào thủ tục.
- Hàm tính lương biết ngày công và chức vụ.
- Hàm tính thưởng biết số ngày công.

http://tieulun.hopto.org

Vận dụng thủ tục và hàm nói trên này để thực hiện các thủ tục dưới đây:

- Thủ tục nhập dữ liệu: nhập n và thông tin cho n nhân viên.
- Thủ tục in lên màn hình lương tổng cộng của từng nhân viên.

Bài 4.7

Viết chương trình trong đó có các hàm sau đây:

- Hàm tính độ dài đoạn thẳng đi qua 2 điểm (x_1,y_1) và (x_2,y_2).
- Hàm tính diện tích tam giác mà 3 cạnh là a,b,c.

Vận dụng các hàm nói trên để tính và in lên màn hình diện tích tứ giác lồi biết tọa độ 4 đỉnh của nó nhập từ bàn phím. Giả thiết dữ liệu nhập vào là thỏa mãn tính chất lồi của tứ giác, không cần kiểm tra.

Bài 4.8*

Viết chương trình nhập từ bàn phím tọa độ n điểm có toạ độ nguyên trong mặt phẳng của một đường gấp khúc đi qua n điểm đó. Chương trình cần cho biết đường gấp khúc này có tự cắt hay không.

Bài 4.9*

Viết chương trình nhập từ bàn phím toạ độ nguyên của 4 đỉnh của một tứ giác. Kiểm tra và thông báo lên màn hình một trong các tính chất sau đây:

- Tứ giác đó suy biến theo nghĩa không phải là tứ giác.
- Tứ giác đó lồi.
- Tứ giác đó lõm.

Bài 4.10*

Các phân số nói trong bài tập này có dạng x/y trong đó x,y là các số nguyên và x,y có ước chung duy nhất là 1. Viết chương trình nhập từ bàn phím 2 phân số a và b. In lên màn hình các phân số là kết quả của:

1. a + b
2. $a^2 - b^2$
3. $(a+b)^3$

http://tieulun.hopto.org

Chương 5

DỮ LIỆU KIỂU TỆP (FILE)

Đặt vấn đề

Khi thực sự giải quyết các bài toán thực tế thì dữ liệu vào và dữ liệu ra thường khá lớn và được ghi vào tệp trên bộ nhớ ngoài, ví dụ như đĩa. Độ lớn của dữ liệu càng lớn thì càng đánh giá được tính khả thi của một thuật toán xét về mặt thời gian chạy chương trình.

Có 3 loại tệp: tệp văn bản, tệp định kiểu và tệp không kiểu.

Trong chương này chúng ta không quan tâm đến cách tổ chức vật lý của tệp mà chỉ quan tâm đến mức trên cùng là mức ứng dụng của các tệp, nghĩa là biết cách sử dụng tệp với tư cách là công cụ để vào/ra dữ liệu cho chương trình, còn cái mà chúng ta quan tâm ở trong chương trình trước sau vẫn là thuật toán.

Tệp văn bản được quan tâm hơn cả bởi vì tệp văn bản thích hợp cho việc **xem, sửa** dữ liệu vào/ra.

Tệp định kiểu thích hợp cho các bài toán mô phỏng quản trị cơ sở dữ liệu. Tuy nhiên, việc tổ chức và thao tác trên dãy mẫu tin cùng kiểu có lẽ thích hợp hơn đối với các phần mềm quản trị CSDL thật sự. Ưu điểm hơn tệp định kiểu, tệp văn bản cho phép chúng ta nhìn thấy được bằng mắt dữ liệu mà thuật toán đang xử lý: bao gồm dữ liệu vào, dữ liệu ra và kể cả các kết quả trung gian trong quá trình tính toán (nếu như chúng ta muốn quan tâm đến các kết quả trung gian này, chẳng hạn để từng bước theo dõi thuật toán).

Tệp không định kiểu do vậy cũng không cần thiết nêu ra ở đây nếu thật sự chúng ta không muốn đi chi tiết vào ngôn ngữ hơn là thuật toán.

http://tieulun.hopto.org

I. TỆP VĂN BẢN

1. Tệp văn bản ngầm định INPUT và OUTPUT

1.1. Khai báo biến tệp văn bản

VAR tên_biến_tệp: TEXT;

Ví dụ:

Var f: text;

Trong đó f là là tên biến tệp dùng để xác định một tệp văn bản nào đó trên đĩa. Tên của tệp này sẽ được chỉ định trong lệnh "gán" tên tệp cho biến tệp: Assign(f,tên_tệp);

Ví dụ:

Assign(f,'C:\TP\INPUT.TXT');

Nhiều khi, đáng lý phải nói "tệp input.txt", để cho ngắn gọn ta thường nói "tệp f ".

1.2. Tệp văn bản ngầm định OUTPUT

Để dễ hiểu, ta có thể cho rằng các thao tác vào/ra trên tệp văn bản cũng giống như thao tác vào/ra trên màn hình. Chẳng hạn để ghi lên dòng hiện tại của màn hình một số thực x với lệnh:

writeln(x:10:3);

thì khi lệnh ghi số thực đó vào dòng hiện tại của tệp f sẽ là:

writeln(f,x:10:3);

Như vậy lệnh ghi dữ liệu ra tệp văn bản chỉ khác lệnh ghi dữ liệu lên màn hình ở chỗ: trong lệnh ghi dữ liệu ra tệp văn bản có chỉ định tên biến tệp f trước khi chỉ ra các dữ liệu cần ghi. Kết quả hiện ra trong tệp văn bản cũng như trên màn hình, nghĩa là số thực x chiếm 10 vị trí trong đó phần thập phân chiếm 3 vị trí, tất cả dữ liệu được canh biên phải.

Lệnh:

writeln(x:10:3)

thực ra là lệnh:

writeln(OUTPUT,x:10:3)

trong đó OUTPUT mặc định là tên biến tệp văn bản chuẩn để chỉ màn hình (một thiết bị ra chuẩn). Và khi đã là mặc định thì tên của biến tệp OUTPUT có thể không cần chỉ ra trong lệnh.

http://tieulun.hopto.org

Vậy, một cách tổng quát, tên biến tệp văn bản có thể là:

- OUTPUT để chỉ màn hình,
- LST để chỉ máy in.
- Một tên biến tệp, ví dụ là f để chỉ tệp văn bản (Kiểu TEXT).

1.3. Tệp văn bản ngầm định INPUT

Nếu màn hình là tệp văn bản mà tên biến tệp mặc định là OUTPUT - để xuất dữ liệu thì bàn phím là tệp văn bản có tên biến tệp mặc định là INPUT - để nhập dữ liệu.

Vậy lệnh:

Readln(f,biến$_1$,...,biến$_n$)

trong đó f kiểu text hoặc thay bằng tên biến INPUT để chỉ bàn phím.

Vậy có thể định nghĩa theo quan điểm lập trình rằng "Màn hình là một tệp văn bản chuẩn, máy in là một tệp văn bản. Bàn phím là một tệp văn bản chuẩn".

2. Cách sử dụng tệp văn bản

Mở tệp văn bản để ghi dữ liệu từ biến vào tệp, gọi tắt là mở tệp để ghi.

Mở tệp văn bản để đọc dữ liệu từ tệp ra biến, gọi tắt là mở tệp để đọc.

Sử dụng tệp văn bản để ghi khác với sử dụng tệp để đọc. Cách sử dụng được tóm tắt như bảng dưới đây:

Giả thiết có khai báo:

```
Const
fname = 'Tên_tệp_viết_giữa_2_dấu_nháy_đơn' ;
Var f: text;
```

Khi đó ta có:

Mở tệp để ghi	Mở tệp để đọc
Bước 1: Mở tệp *Assign(f,fname); Rewrite(f);*	*Bước 1: Mở tệp* *Assign(f,fname); Reset(f);*
Bước 2: Ghi tệp: Dùng các lệnh sau: *Write(f, bthức1, bthức2, ...);* *Writeln(f, bthức1, bthức2, ...);* *Writeln(f);*	*Bước 2: Đọc tệp: Dùng các lệnh sau:* *Read(f, biến1, biến2, ...);* *Readln(f, biến1, biến2, ...);* *Read(f);*

http://tieulun.hopto.org

Bước 3: Đóng Tệp *Close(f);*	*Bước 3: Đóng Tệp* *Close(f);*
	Các hàm hỗ trợ EOF(f): Kiểm tra hết tệp. EOLN(f): Kiểm tra cuối dòng. SeekEoln(f), SeekEof(f).

2.1. Mở tệp để ghi

2.1.1. Lệnh mở tệp để ghi

Lệnh Rewrite(f,fname): mở tệp có tên tệp (với đường dẫn đầy đủ) được chỉ ra trực tiếp giữa 2 dấu nháy hoặc chứa trong hằng/biến xâu ký tự fname.

Tác dụng của lệnh này là: luôn luôn tạo ra một tệp văn bản mới. Trong trường hợp trên đĩa đã có tệp fname thì tệp fname này bị xóa đi để tạo mới từ đầu. Việc ghi dữ liệu vào tệp fname bắt đầu từ dòng đầu tiên.

Nếu trên đĩa đã có tệp fname và ta không muốn xóa nội dung đã có để tiếp tục bổ sung các dòng dữ liệu mới vào cuối tệp thì ta thay lệnh Rewrite(f) bằng lệnh: Append(f);

Tuy nhiên, đôi lúc ta vẫn sử dụng lệnh Append(f); trong khi ta quên mất rằng trên đĩa chưa có tệp fname. Trong trường hợp này, lúc chạy, chương trình sẽ báo lỗi: "File not found". Để chủ động trong việc xử lý lỗi ta có thể dẫn hướng biên dịch để bật tắt kiểm tra vào/ra hợp lý như dưới đây:

```
Assign(f,fname);
{$I-}
  Append(f);
{$I+}
If IOresult > 0 Then
Begin
  Writeln('Chua co tep ',fname,' tren dia. Tep moi duoc tao tu dau');
  Rewrite(f);
End;
```

Trong đó, hàm IOResult là hàm nhận kết quả mở tệp bằng lệnh Append(f) hoặc Reset(f). Khi các lệnh này thực hiện thành công thì hàm trả lại giá trị 0, khi các lệnh này thực hiện có lỗi, thì hàm trả lại mã lỗi khác 0.

http://tieulun.hopto.org

2.1.2. Các lệnh ghi dữ liệu vào tệp

Một khái niệm (đối tượng) đặc biệt dùng để xác định vị trí truy xuất tệp là con trỏ tệp. Khi bắt đầu mở tệp thì con trỏ tệp trỏ vào phần tử dữ liệu đầu tiên của dòng đầu tiên. Trong quá trình ghi/đọc dữ liệu, con trỏ tệp sẽ trỏ vào các phần tử dữ liệu trên một dòng xác định hoặc chuyển xuống dòng tiếp theo để tiếp tục đợi lệnh truy xuất dữ liệu mới. Thuật ngữ "dòng hiện tại của tệp..." nghĩa là muốn nói rằng con trỏ tệp đang trỏ vào dòng đó.

Lệnh Write(f, bthức1, bthức2,..... bthức_n); có tác dụng ghi giá trị của n biểu thức vào dòng hiện tại của tệp. Sau khi ghi xong n phần tử dữ liệu thì con trỏ tệp nằm ở sau phần tử dữ liệu thứ n, thường là cuối dòng hiện tại.

Lệnh Writeln(f, bthức1, bthức2,..... bthức_n); có tác dụng giống như lệnh Write(f,...) nhưng sau khi ghi xong n phần tử dữ liệu thì con trỏ tệp chuyển xuống đầu dòng dưới, để sẵn sàng ghi dữ liệu vào dòng mới này.

Việc ghi giá trị của nhiều biểu thức vào tệp văn bản, cũng như khi ghi lên màn hình, dữ liệu có thể được in ra có quy cách và giữa các mục dữ liệu có thể có thêm các ký tự ngăn cách, chẳng hạn như dấu cách. Ví dụ để in vào dòng hiện tại các số nguyên i,j,k, có thể viết lệnh:

Writeln(i,' ',j,' ',k);

Lệnh Writeln(f); ghi dữ liệu rỗng vào dòng hiện tại và chuyển con trỏ tệp xuống đầu dòng dưới. Thường thì lệnh Writeln(f) được sử dụng để trình bầy các khối dữ liệu mà các khối được ngăn cách bằng một vài dòng trống, mỗi một dòng trống được tạo bởi một lệnh Writeln(f).

2.1.3. Lệnh đóng tệp

Lệnh Close(f); có tác dụng đóng tệp văn bản, đẩy dữ liệu của tệp trong bộ nhớ lên đĩa và kết thúc công việc truy xuất dữ liệu trên tệp.

Ví dụ 5.1

Hãy tạo tệp PRIMES.TXT chứa các số nguyên tố trong đoạn từ 1..20.000. Mỗi dòng của tệp chứa 20 số, trừ dòng cuối cùng có thể ít hơn.

PRIMES.PAS

```
const fname = 'PRIMES.TXT';
var
a,d,i,s: integer;
Begin
Assign(f,fname); Rewrite(f);
```


http://tieulun.hopto.org

```
s:=0; {so luong cac so nguyen to}
for a:=2 to 20000 do
begin
d:=0; {so luong cac uoc cua a }
for i:=1 to a div 2 do
if a mod i = 0 then inc(d);
if d=1 then
begin
     inc(s);
     if s mod 20 = 0 then writeln(f,a)
     else write(f,a,' ');
end;
end;
close(f);
End.
```

2.2. Mở tệp văn bản để đọc

2.2.1. Lệnh mở tệp để đọc

Lệnh Reset(f): là lệnh mở tệp f đã tồn tại trên đĩa. Cũng giống như lệnh Append(f), nếu trên đĩa không có tệp fname thì khi chạy, chương trình sẽ báo lỗi "File not found'. Ta có thể dẫn hướng biên dịch để bật tắt kiểm tra vào ra hợp lý như sau:

```
Assign(f,fname);
{$I-} Reset(f); {$I+}
If IOresult > 0 Then
Begin
  Writeln('Chua co tep ',fname, 'tren dia. Dung thuc
          hien chuong trinh');
  Readln; Halt;
End;
```

2.2.2. Các lệnh đọc dữ liệu

Lệnh Read(f, biến1, biến2,....biến_n): có tác dụng đọc lần lượt n phần tử dữ liệu trên dòng hiện tại của tệp để gán cho n biến tương ứng theo thứ tự đã

http://tieulun.hopto.org

được chỉ ra trong lệnh. Sau khi đọc xong n phần tử dữ liệu thì con trỏ tệp nằm ở sau phần tử dữ liệu thứ n, thường là cuối dòng hiện tại.

Lệnh Readln(f, biến1, biến2,....., biến_n); có tác dụng giống như lệnh Read(...) nhưng sau khi đọc xong n phần tử dữ liệu ra n biến thì con trỏ tệp chuyển xuống đầu dòng dưới, để sẵn sàng đọc dữ liệu ở dòng dưới này.

Lệnh Readln(f): vẫn đọc dòng hiện tại nhưng không đọc dữ liệu ra biến nào mà chuyển con trỏ tệp xuống dòng dưới. Thường thì lệnh Readln(f) được sử dụng để đọc "vượt qua" những dòng không có dữ liệu mà trước đó đã được ghi bằng lệnh writeln(f);

2.2.3. Sử dụng các hàm về tệp

Việc đọc dữ liệu trên tệp văn bản thường kết hợp với các hàm eof(f), eoln(f), seekeof(f), seekeoln(f).

Hàm EOF(f) kiểu Boolean (end of file). Hàm có giá trị TRUE khi con trỏ tệp đã chỉ vào sau dòng dữ liệu cuối cùng của tệp. Hàm có giá trị bằng FALSE trong trường hợp ngược lại.

Hàm EOLN(f) kiểu Boolean (end of line). Hàm có giá trị TRUE khi con trỏ tệp đã chỉ vào cuối dòng dữ liệu hiện tại của tệp. Hàm có giá trị bằng FALSE trong trường hợp ngược lại.

Hàm SeekEoln(f) kiểu Boolean. Hàm tương tự như hàm Eoln(f) nhưng trước khi kiểm tra dấu hiệu hết dòng, con trỏ tệp nhảy qua các khoảng cách ghi bởi các ký tự Space và Tab.

Hàm SeekEof(f) kiểu Boolean. Hàm tương tự như hàm Eof(f) nhưng trước khi kiểm tra dấu hiệu hết tệp, con trỏ tệp nhảy qua các khoảng cách và các dòng trống ghi bởi các ký tự Space, Tab và CR-LF.

Ví dụ 5.2

Hãy đọc các số nguyên tố từ tệp PRIMES.TXT và ghi chúng lên màn hình. Về mặt kỹ thuật: Mỗi dòng màn hình ghi không quá 10 số, có tạm dừng màn hình mỗi khi hiển thị được 24 dòng.

PRIMES2.PAS

```
const fname = 'PRIMES.TXT';
var
f: text;
a,s: integer;
Begin
```


http://tieulun.hopto.org

```
Assign(f,fname); Reset(f);
s:=0; {so luong cac so nguyen to}
while not EOF(f) do
begin
while not EOLN(f) do
begin
  read(f,a); inc(s);
  if s mod 10 = 0 then writeln(a) else write(a,' ');
  if s mod 24 = 0 then readln;
end;
Readln(f);
end;
Close(f);
End.
```

2.3. Đặc điểm của tệp văn bản

Khi ghi dữ liệu vào tệp văn bản thì tất cả các kiểu dữ liệu được tự động chuyển thành kiểu ký tự. Ngược lại, khi đọc dữ liệu ra các biến thì các phần tử dữ liệu sẽ tự động "khôi phục" lại các kiểu dữ liệu như lúc ghi vào để gán cho các biến. Tất nhiên các biến, theo thứ tự, cũng phải có kiểu dữ liệu tương ứng giống với các kiểu dữ liệu của các phần tử dữ liệu đang được đọc.

Có một ngoại lệ là có thể đọc tất cả một dòng dữ liệu (thuộc các kiểu khác nhau) ra một biến xâu ký tự và biến xâu ký tự này sẽ nhận giá trị là toàn bộ nội dung dòng ký tự vừa được đọc đó.

Ví dụ 1

Nếu ta có khai báo:

```
Var
    m,n: integer;
    r : real;
    c : char;
    s : string;
```

và gán:

```
    m:=10;
    n:=20;
    r:=9.51;
```


http://tieulun.hopto.org

c:='A';

Thì lệnh:

Writeln(f,m,' ',r:4:2,' ',c,' ',n);

sẽ ghi vào tệp f một xâu ký tự có dạng:

10 9.51 A 20

Bây giờ nếu con trỏ tệp f đang trỏ vào đầu dòng văn bản trên:

Nếu đọc dòng này bằng lệnh:

Readln(f,m,r,c,n);

ta sẽ được kết quả m,n,r,c như ban đầu.

Nếu ta đọc dòng này bằng lệnh:

Readln(f,s);

ta sẽ được xâu s có giá trị là:

s='10 9.51 A 20'

Một điều rất chú ý là các phần tử dữ liệu thuộc các kiểu khác nhau, trừ kiểu String, được ngăn cách bởi ít nhất một dấu cách (ký tự #32). Riêng đối với một phần tử dữ liệu thuộc kiểu String ta chỉ nên ghi trọn vẹn trên một dòng. Có nghĩa là lệnh Read[ln] không phân biệt được 2 mục dữ liệu khác nhau nếu chúng đều có kiểu string.

Ví dụ 2

Nếu ta có:

Var x,y: string;

x:='Thu Ha';

y:='Chi Dung';

Thì lệnh:

Writeln(f,x,' ',y);

sẽ ghi vào tệp một dòng văn bản là:

Thu Ha Chi Dung

Bây giờ nếu đọc dòng này bằng lệnh:

Readln(f,x,y);

Thì kết quả

x='Thu Ha Chi Dung';

y='';

http://tieulun.hopto.org

Một cách bản chất: Các phần tử của tệp văn bản là các ký tự và được tổ chức thành các dòng. Các dòng ngăn cách bằng các ký tự hết dòng, đó là cặp ký tự CR,LF (CR: Carriage Return: về đầu dòng, LF: Line Feed: chuyển xuống dòng tiếp theo. Mã ASCII của CR là 13, của LF là 10).

Ví dụ 2 lệnh:

Writeln(f,x); Writeln(f,y);

Có thể thay bằng một lệnh:

Writeln(f,x,#13,#10,y);

3. Một số kỹ năng sử dụng tệp văn bản

Như cách đặt vấn đề ban đầu, tệp văn bản có vai trò như công cụ để vào/ra dữ liệu cho nên có thể nói là tất cả các bài tập và nhất là các bài toán có dữ liệu thực tế đều sử dụng tệp văn bản để chứa dữ liệu vào và dữ liệu ra của chương trình.

Vì lý do trên, không thể phân dạng bài tập về tệp văn bản giống như khi ta nghiên cứu các kiểu dữ liệu có cấu trúc như mảng, xâu ký tự, tập hợp, bản ghi... Một lần nữa có thể nhấn mạnh rằng tệp văn bản chỉ thay thế cho việc nhập dữ liệu bằng tay từ bàn phím và xem kết quả trên màn hình.

Từ đó, ưu điểm nổi bật của việc sử dụng tệp văn bản là quá trình vào/ra dữ liệu, khi dữ liệu rất lớn, ta không phải mất công sức nhập dữ liệu nhiều lần mỗi khi chạy kiểm định chương trình. Dữ liệu vào đã được lưu trên tệp trên đĩa để chương trình sử dụng cho mỗi một lần thực hiện. Dữ liệu ra cho dù rất dài cũng đã được lưu trên tệp để xem, sửa, in ấn và sao chép giữa các máy tính...

Trong mục này, chúng ta sẽ đưa ra một vài kinh nghiệm về mặt tổ chức chương trình có sử dụng tệp văn bản làm công cụ vào/ra và coi đó là một số kỹ năng sử dụng tệp văn bản.

3.1. Trường hợp 1: Input (bàn phím), Output (tệp văn bản)

Mẫu 1:

```
const foup = 'tên_tệp_ra';
Var f: text;
BEGIN
  <Nhập dữ liệu từ bàn phím cho các biến>;
  Assign(f,foup); Rewrite(f);
  <Giải quyết bài toán, xử lý tính toán trên các biến>;
```


http://tieulun.hopto.org

```
    <Ghi dữ liệu kết quả từ các biến vào file f>;
    Close(f);
  END.
```

Ví dụ 5.3

Cặp số (a,b) gọi là cặp số hoàn thiện nếu tổng các ước (khác chính nó) của số này bằng số kia và ngược lại. Ví dụ , các cặp số sau là hoàn thiện:

(6,6), (28,28), (220,284), (1184,1210), (2620,2924),...

Vì các cặp số hoàn thiện khá hiếm và có nhiều tính chất quan trọng nên người ta muốn lập trình tạo sẵn các cặp số hoàn thiện để tiện sử dụng.

Hãy viết chương trình tạo tệp văn bản có tên là HOANTHIEN.DAT chứa các cặp số hoàn thiện phân biệt lấy trong đoạn [p..q], p, q là các số nguyên dương nhập từ bàn phím. Mỗi cặp số hoàn thiện được ghi trên một dòng riêng biệt. Riêng dòng cuối cùng của tệp ghi số lượng các cặp số hoàn thiện tìm được.

Bài làm:

```
Program Bai1;
Const fout     =     'HOANTHIEN.DAT';
Var
   f        :     text;
   i,a,b,s,p,q,d:     integer;
(*-------------------------------------*)
BEGIN
 Repeat
   Write('p, q = '); Readln(p,q);
 Until (p>0) and (p<q);
 d:=0;
 Assign(f,fout); Rewrite(f);
 for a:=p to q do
 begin
  b:=0;
  for i:=1 to a div 2 do
```


http://tieulun.hopto.org

```
if a mod i = 0 then b:= b + i;
s:=0;
for i:=1 to b div 2 do
if b mod i = 0 then s:= s + i;
if (s=a)and(a<=b) then
begin
  Writeln(f,a,' ',b);
  inc(d);
 end;
end;
Writeln(f,d);
Close(f);
END.
```

3.2. Trường hợp 2: Input (tệp văn bản), Output (màn hình)

Mẫu 2:

```
const finp = 'tên_tệp_vào';
Var f: text;
BEGIN
 Assign(f,finp); Reset(f);
 <Đọc dữ liệu từ file f ra các biến>;
 Close(f);

 <Giải quyết bài toán, xử lý tính toán trên các biến>;
 <In dữ liệu kết quả lên màn hình>;.
END.
```

Ví dụ 5.4

Giả thiết trên đĩa đã có file văn bản "QLCB.DAT" lưu trữ một danh sách không quá 100 cán bộ. Thông tin của mỗi cán bộ được ghi thành một bộ dữ liệu gồm 3 dòng, mỗi dòng lần lượt ghi: Họ đệm, Tên, Lương của cán bộ đó. Các bộ dữ liệu cách nhau một dòng trống (Nếu soạn thảo bằng tay để tạo ra tệp QLCB.DAT cũng phải tuân theo nguyên tắc đó).

http://tieulun.hopto.org

Hãy viết chương trình hiển thị lên màn hình danh sách các cán bộ (gồm họ tên đầy đủ và lương) theo thứ tự abc của mục Tên.

Bài làm:

```
Program Bai2;
Uses crt;
Const finp     =     'QLCB.DAT';
Type  NHANSU =     Record
     Hodem: String[20];
     Ten : String[9];
     Luong: Integer;
     End;
Var
    CB : Array[1..100] Of NHANSU;
    n : Integer;
    f : text;
Procedure ReadFile;
Var
    i: integer;
Begin
    Assign(f,finp); Reset(f);
    n:= 0;
    while not eof(f) do
    begin
     inc(n);
     Readln(f,CB[n].hodem);
     Readln(f,CB[n].ten);
     Readln(f,CB[n].luong);
     Readln(f);
    end;
    Close(f);
End;
```


http://tieulun.hopto.org

```
Procedure Sorting;
Var
     Temp : NHANSU;
     i,j : integer;
Begin
 For i:=2 to n do
 For j:=n downto i do
 if CB[j].Ten < CB[j-1].Ten then
 Begin
    TemP:= CB[j];
    CB[j]:= CB[j-1];
    CB[j-1]:= Temp;
 End;
End;
Procedure Listing;
Var
   i: integer;
Begin
 Writeln('Danh sach CB sap xep theo abc cua ten la: ');
 For i:=1 to n do Writeln(CB[i].Hodem,' ',CB[i].Ten);
 Readln;
End;
Begin
 ReadFile;
 Sorting;
 Listing;
 Readln;
End
```

3.3. Trường hợp 3: Input (tệp văn bản), Output (tệp văn bản)

Mẫu 3: Vào, ra, xử lý độc lập

```
const finp = 'tên_tệp_vào';
      foup = 'tên_tệp_ra';
```


http://tieulun.hopto.org

```
Var f: text;
BEGIN
  Assign(f finp); Reset(f);
   <Đọc dữ liệu từ file f ra các biến>;
  Close(f);
 Assign(f foup); Rewrite(f);
    <Giải quyết bài toán, xử lý tính toán trên các biến>;
    <Ghi dữ liệu kết quả từ các biến vào file f>;
    Close(f);
    END.
```

Ví dụ 5.5

Cho dãy n số nguyên a1, a2,..., a3. (n <= 1000, ai <= 30000, i =1,2,...n)

Viết chương trình:

1) Sắp xếp dãy số trên theo thứ tự tăng dần.

2) Tìm các phần tử là số nguyên tố trong dãy đã cho.

Dữ liệu vào cho trong tệp văn bản có tên là DAYSO.INP, có nội dung như sau:

- Dòng 1 là 1 số nguyên n.
- Dòng 2 gồm n số nguyên biểu diễn các phần tử của dãy số nói trên.

Dữ liệu ra ghi vào một tệp văn bản có tên là DAYSO.OUT:

- Dòng 1: chứa dãy số nói trên đã được sắp xếp.
- Dòng 2: giữ nguyên thứ tự trước sau của các phần tử trong dãy số ban đầu (chưa được sắp xếp), để ghi vào tệp các phần tử là số nguyên tố, các phần tử khác ghi vào tệp ký tự 'x'.

Ví dụ:

DAYSO.INP	DAYSO.OUT
8 3 12 81 13 6 5 65 29	3 5 6 12 13 29 65 81 3 x x 13 x 5 x 29

Bài làm:

```
Program Baitap3;
Const
   finp     =     'Dayso.inp';
```


http://tieulun.hopto.org

```
    fout      =     'Dayso.out';
    maxn      =     1000;
    maxvalue  =     10000;
Type
  Arr =     array[1..maxn] of integer;
Var
     f      :     text;
     a      :     Arr;
     n      :     integer;
(*-------------------------------------*)
Procedure Doc_Tep;
Var f: text; i: integer;
Begin
 Assign(f,finp); {$I-} Reset(f); {$I+}
 If IoResult>0 Then
 Begin  Write('Khong co file ',finp,' tren dia ');
      Readln; Halt;
 End;
 Readln(f,n);
 For i:=1 to n do read(f,a[i]);
 close(f);
End;
(*-------------------------------------*)
Procedure Sap_xep(a: Arr);
Var i,j, k,tg : integer;
Begin
    for i:=1 to n-1 do
    begin k:=i;
        for j:=i+1 to n do
        if a[j]<a[k] then k:=j;
        If k <> i Then
        begin tg:= a[i];
            a[i]:=a[k];
```


http://tieulun.hopto.org

```
            a[k]:=tg;
        end;
    end;
    For i:=1 to n do write(f,a[i]:5);
    Writeln(f);

End;
(*-----------------------------------*)
Function Nguyen_to(x: integer): Boolean;
Var d,j: integer;
Begin
    d:=0;
    for j:=1 to x div 2 do
    if x mod j = 0 then d:=d + 1;
    Nguyen_to:= (d=1);
End;
(*-----------------------------------*)
Procedure Tim_nguyen_to;
Var i: integer;
Begin
   for i:=1 to n do
   If nguyen_to(a[i]) then Write(f,a[i]:5)
   Else Write(f,'*':5);
End;
(*-----------------------------------*)
BEGIN
    Doc_tep;
    Assign(f,fout); Rewrite(f);
     Sap_xep(a);
     Tim_nguyen_to;
   Close(f);
END.
```


http://tieulun.hopto.org

Mẫu 4: Vào, ra, xử lý đồng thời

```
const finp = 'tên_tệp_vào';
    foup = 'tên_tệp_ra';
Var f1,f2: text;
BEGIN
 Assign(f1,finp); Reset(f1);
 Assign(f2,foup); Rewrite(f2);
 While Not Eof(f1) do
 Begin
  <Đọc dữ liệu từ file f1 ra các biến>;
 <Giải quyết bài toán, xử lý tính toán trên các biến>;
  <Ghi dữ liệu kết quả từ các biến vào file f2>;
 End;
 Close(f1); Close(f2);
END.
```

Ví dụ 5.6

Nếu theo đúng cú pháp của câu văn thì phải viết dấu phẩy rồi mới viết một dấu cách, tương tự viết dấu chấm rồi mới viết một dấu cách. Hãy viết chương trình soát và sửa lỗi chính tả của một đoạn văn xét về phương diện các dấu ngăn cách các thành phần trong câu (dấu phẩy và ngăn cách giữa các câu (dấu chấm).

Văn bản bị lỗi chứa trong tệp XULYVB.INP.

Văn bản sau khi được sửa ghi vào tệp XULYVB.OUT.

Chú ý rằng một từ trong văn bản giả thiết không bị ngắt giữa chừng để xuống dòng, dấu ngăn cách không xuất hiện ở đầu dòng và chương trình không cần quan tâm đến các lỗi chính tả dạng khác mà không đề cập đến trong bài này.

Ví dụ (Để cho dễ nhìn, trong ví dụ này, một số dấu cách được thay thế bằng ký tự *):

XULYVB.INP

Cac kieu du lieu don gian trong Pascal la integer*,real,**char,boolean va string*.*Cac kieu du lieu co cau truc ma ta da hoc la array**,**record*,file*.Kieu string cung co the coi la kieu du lieu nua cau truc.

http://tieulun.hopto.org

XULYVB.OUT

Cac kieu du lieu don gian trong Pascal la integer,*real,*char,*boolean va string.*Cac kieu du lieu co cau truc ma ta da hoc la array,*record,*file.*Kieu string cung co the coi la kieu du lieu nua cau truc.

Bài làm:

```
Program Baitap4;
Const
   finp     =     'XULYVB.INP';
   fout     =     'XULYVB.OUT';
   space    =     ' ';
   phay     =     ',' ;
   cham     =     '.';
Var  f1,f2     :     text;
   s        :     string;
(*-------------------------------------*)
Procedure Sua_loi_dau(dau:char);
Var i,k   :    Byte;
Begin
  i:=1; { Vi du dau la dau phay }
  Repeat { Them 1 dau cach sau dau phay}
   if s[i]=dau then insert(space,s,i+1);
   i:=i+1;
  Until i>= length(s);
  Repeat { Xoa cac dau cach truoc dau phay }
    k:= pos(space+dau,s);
    If k >0 Then Delete(s,k,1);
  Until k=0;
  Repeat { xoa dau cach thua sau day phay }
   k:= pos(dau+space+space,s);
   If k > 0 Then Delete(s,k+1,1);
  Until k=0;
```


http://tieulun.hopto.org

```
 End;
BEGIN
 Assign(f1,finp); Reset(f1);
 Assign(f2,fout); Rewrite(f2);
 While not EOF(f1) do
 Begin
  Readln(f1,s);
  Sua_loi_dau(phay);
  Sua_loi_dau(cham);
  Writeln(f2,s);
 End;
 Close(f1); Close(f2);
END.
```

4. Câu hỏi và bài tập

Bài 5.1

Viết chương trình trong đó có các hàm sau đây:

+ Isprime(n): Cho kết quả True nếu n là số nguyên tố và cho False nếu n không phải số nguyên tố.

+ Hàm ListPrime(n): Liệt kê các số nguyên tố từ 2 đến n.

Áp dụng các hàm nói trên, chương trình cần liệt kê các số nguyên tố trong đoạn [2..N].

Dữ liệu vào cho trong file văn bản "LIST_N.INP", mỗi dòng ghi một số nguyên N <= 10000.

Tương ứng với mỗi dòng của file dữ liệu vào, dãy các số nguyên tố tìm được theo yêu cầu đầu bài ghi trên một dòng riêng biệt của file dữ liệu ra là một file văn bản có tên là " PRIMES.OUT".

Bài 5.2

Giả sử mỗi cán bộ có các thông tin sau: Họ đệm, Tên, Tuổi.

Hãy viết chương trình sử dụng một file văn bản hoặc file định kiểu có tên là "QLNS.DAT" để lưu thông tin về các cán bộ. Chương trình tạo ra một menu gồm các chức năng sau đây:

http://tieulun.hopto.org

+ Hiển thị thông tin về các cán bộ lên màn hình.

+ Hiển thị lên màn hình họ tên đầy đủ và tuổi của các cán bộ có cùng tên S (S nhập từ bàn phím).

+ Thoát khỏi chương trình.

Bài 5.3

Cho tệp văn bản có tên là MESSAGE.INP ghi nội dung của một tài liệu. Giả thiết rằng: Giữa 2 câu chỉ ngăn cách bởi một dấu chấm và các từ trong câu chỉ ngăn cách bằng dấu trắng (dấu cách) hoặc dấu phẩy.

Mỗi dòng chứa không quá 200 ký tự và có không quá 100 dòng trong tệp. Tệp MESSAGE.INP chỉ chứa văn bản đơn thuần (không bao gồm các hình vẽ hay biểu bảng).

Viết chương trình đọc tệp văn bản nói trên và tạo tệp văn bản MESSAGE.OUT để ghi các thông tin sau đây:

1) Dòng 1 ghi số lượng các từ có trong tệp message.inp.

2) Dòng 2 ghi lại một câu có nhiều từ nhất trong tệp message.inp.

Ví dụ

MESSAGE.INP

Children need a good care.

Without any explanation, everybody knows that children need a good care. In another way, all the people should take much care of children.

MESSAGE.OUT

28

In another way, all the people should take much care of children.

Bài 5.4

Cho một tệp văn bản có tên NUMBERS.INP có nội dung:

- Dòng 1 chứa 1 số nguyên n (n <=100).
- Dòng 2 chứa dãy n số nguyên dương a_1 a_2... a_n (a_i <= 1000, i =1,2,...n)

Viết chương trình đọc tệp NUMBERS.INP và:

1) Tìm một phần tử lớn nhất trong dãy số đã cho.

2) Tìm ước số chung lớn nhất của dãy số nói trên.

Kết quả tìm được in lên màn hình.

http://tieulun.hopto.org

Ví dụ:

DAYSO.INP	Kết quả in lên màn hình
10 300 84 90 192 438 198 594 96 234 564	Max = 594 USCLN = 6

Bài 5.5

Các công cụ trực quan của các phần mềm quản trị cơ sở dữ liệu như Access, Visual Fox, Visual Basic không có khả năng xử lý chi tiết dữ liệu dạng chuỗi ký tự và cần có sự can thiệp của lập trình.

Ví dụ cần phải liệt kê danh sách các sinh viên có tên là "Nguyen*Chi*Dung" (ký hiệu * thay cho dấu cách để dễ quan sát trong đề bài). Khi đó các phần mềm trên sẽ bỏ qua các sinh viên "**Nguyen*Chi*Dung*", "*Nguyen**Chi***Dung"... mặc dù họ đều là các sinh viên đang được quan tâm.

Sự can thiệp của chương trình ở đây là làm chuẩn hoá họ tên của các sinh viên sao cho không có các dấu cách ở đầu, không có quá 1 dấu cách giữa họ, đệm và tên.

Giả sử trên đĩa đã cho tệp DSSV.TXT mỗi dòng chứa họ tên của một sinh viên. Hãy viết chương trình Pascal tạo tệp DSSV2.TXT ghi lại họ tên của sinh viên đó nhưng các họ tên này đã được chuẩn hoá.

II. TỆP ĐỊNH KIỂU

1. Định nghĩa tệp định kiểu

1.1. Sơ lược về khái niệm tệp

Tệp (file) là một đơn vị thông tin lưu trữ trên bộ nhớ ngoài, ví dụ như đĩa từ. Nói cách khác, mọi chương trình, dữ liệu, số liệu đều được ghi trên đĩa dưới dạng một tệp.

Việc tổ chức lưu trữ tệp ở mức thấp hơn (mức vật lý) được hệ điều hành đảm nhận, ở mức ứng dụng (mức giao diện), người sử dụng chỉ cần quan tâm đến nguyên tắc truy xuất tệp mà không cần quan tâm đến việc hệ điều hành truy xuất thông tin tương ứng như thế nào trên đĩa.

Tệp được đặc trưng bởi các yếu tố: tên, kích thước, các thuộc tính truy nhập (đọc, ghi, ẩn...), các thuộc tính lưu trữ (ngày, thời gian,...).

http://tieulun.hopto.org

Dưới quan điểm của hệ điều hành, tệp gồm 2 loại: tệp chương trình và tệp dữ liệu. Dưới quan điểm của một hệ quản trị cơ sở dữ liệu, tệp dữ liệu được hiểu theo quan niệm của lý thuyết về các mô hình dữ liệu và cũng được phân thành một số loại. Nói chung, trên nền của hệ điều hành, các chương trình ứng dụng tiếp tục đưa ra các phân loại về tệp ở mức nhìn nhận của nó (mức khung nhìn - view).

Dưới quan điểm lập trình, tệp là một kiểu dữ liệu có cấu trúc biểu diễn dữ liệu lưu trên đĩa. Dữ liệu trên tệp được truy xuất thông qua tên biến tệp và tên tệp cùng với quy tắc sử dụng các lệnh đọc/ghi dữ liệu trên tệp. Các lệnh này được quy định bởi ngôn ngữ lập trình đang sử dụng.

1.2. Định nghĩa tệp định kiểu

Tệp định kiểu là một kiểu dữ liệu có cấu trúc biểu thị một dãy có thứ tự các phần tử có cùng một kiểu dữ liệu và được lưu trữ trên đĩa dưới dạng một tệp.

2. Khai báo biến tệp định kiểu

```
Var
tên_biến_tệp: FILE OF Kiểu_phần_tử;
```

Ví dụ 1

Tệp f mà mỗi phần tử là các số nguyên và tệp g mỗi phần tử là các ký tự:

```
Var
f: File of integer;
g: File of char;
```

Ví dụ 2

Tệp f mà mỗi phần tử là một điểm trong mặt phẳng có tọa độ x,y

```
Type
 PointType = Record
  x,y: real;
 End;
Var
f: File of PointType;
```


http://tieulun.hopto.org

Ví dụ 3:

Tệp f, mỗi phần tử ghi thông tin về một sinh viên, bao gồm: họ tên, giới tính, ngày sinh.

```
Type
   Sinhvien = Record
    hoten: string[30];
    gioitinh: Boolean;
    namsinh: integer;
    End;
 Var
   f: File of Sinhvien;
```

3. Cách sử dụng tệp định kiểu

Tệp định kiểu và mảng một chiều tương tự nhau về sự biểu diễn dãy các phần tử cùng loại và có thứ tự.

Có 3 điểm khác nhau cơ bản giữa tệp và mảng một chiều:

- Mảng chỉ tồn tại trong bộ nhớ còn tệp tồn tại trên đĩa. Vậy tệp không bị mất đi khi tắt máy tính.
- Từ đó, phạm vi biểu diễn của mảng rất hạn hẹp, phụ thuộc vào dung lượng bộ nhớ trong của máy tính. Phạm vi biểu diễn của tệp hầu như không hạn chế, phụ thuộc vào dung lượng còn trống trên đĩa từ và khả năng truy xuất *địa chỉ cao* của hệ điều hành cũng như của chương trình dịch đang sử dụng.
- Việc truy nhập tới các phần tử của tệp khác với mảng. Cụ thể, các lệnh mở tệp để đọc/ghi được tóm tắt như bên dưới.

Chú ý rằng, khác với tệp văn bản, tệp định kiểu sau khi mở có thể vừa thực hiện lệnh đọc, vừa thực hiện lệnh ghi. Ngoài ra lệnh đọc/ ghi dữ liệu đối với tệp định kiểu chỉ gồm 2 lệnh read và write (không có readln và writeln). Vì tương tự như mảng nên ngoài hàm EOF, tệp định kiểu còn có thêm các hàm để xác định vị trí của các phần tử trong tệp. Tệp định kiểu không có hàm EOLN, SeekEoln, SeekEof, Append như đối với tệp văn bản.

Để tiện tìm hiểu các vấn đề tiếp theo, ta sử dụng biến tệp sau đây:

f: FILE OF DataType;

http://tieulun.hopto.org

{Giả sử DataType là một kiểu dữ liệu đã định nghĩa trước đó}

3.1. Mở tệp định kiểu

Mở tệp gồm 2 lệnh:

Gán tên tệp cho biến tệp: Assign(f, tên_tệp);

Sau đó, dùng một trong 2 lệnh mở tệp:

Reset(f): để mở tệp đã có trên đĩa.

Rewrite(f): để tạo ra tệp mới hoàn toàn.

Việc xử lý lỗi khi mở tệp bằng lệnh Reset giống như đối với tệp văn bản.

3.2. Đọc / ghi tệp định kiểu

Có một đối tượng đặc biệt gọi là con trỏ tệp. Con trỏ tệp luôn luôn trỏ vào một vị trí của một phần tử nào đó trong tệp, vị trí này gọi là vị trí hiện tại.

Để ghi một phần tử dữ liệu chứa trong biến item kiểu DataType vào vị trí hiện tại của tệp, ta dùng lệnh:

Write(f,item);

Để đọc một phần tử dữ liệu ở vị trí hiện tại của tệp ra biến item kiểu DataType, ta dùng lệnh:

Read(f,item);

3.3. Các hàm và thủ tục hỗ trợ đọc ghi tệp định kiểu

Hàm EOF(f) kiểu Boolean (end of file). Hàm có giá trị TRUE khi con trỏ tệp đã chỉ vào sau phần tử cuối cùng của tệp. Hàm có giá trị bằng FALSE trong trường hợp ngược lại.

Hàm FileSize(f) cho biết tổng số phần tử hiện có trong tệp.

Hàm FilePos(f) cho biết con trỏ tệp đang trỏ vào vị trí của phần tử thứ bao nhiêu trong tệp.

Thủ tục SEEK(f,n) có tác dụng định vị con trỏ tệp đến vị trí thứ n của tệp để sẵn sàng cho phép đọc/ghi dữ liệu ở vị trí thứ n+1. Trong đó 0 <= n <= FileSize(f).

Ví dụ 1: Để thêm một phần tử vào cuối tệp f từ biến Item, ta dùng các lệnh:

Seek(f, FileSize(f)); Write(f,item);

Ví dụ 2: Các lệnh sau đây:

Seek(f,0); Write(f,item);

sẽ ghi giá trị của item đè lên giá trị của phần tử thứ 1 của tệp.

http://tieulun.hopto.org

3.4. Đóng tệp

Cũng giống như tệp văn bản, sau khi kết thúc thao tác trên tệp. cần phải có lệnh đóng tệp bằng lệnh:

Close(f);

3.5. Xóa và đổi tên tệp

Các lệnh xóa và đổi tên tệp sử dụng đối với tệp bất kỳ.

Để xóa tệp có tên chứa trong biến fname:

Assign(f,fname); ERASE(f);

Để đổi tên tệp mà tên chứa trong fname thành tên mới chứa trong biến fnew:

Assign(f.fname); RENAME(f,fnew);

4. Ví dụ việc sử dụng tệp định kiểu

Như đã đặt vấn đề ở đầu chương, dữ liệu kiểu tệp định kiểu thích hợp cho việc lưu trữ và xử lý các mẫu tin (các bản ghi cùng kiểu). Và, nói chung, các bài toán này cố gắng mô phỏng các chức năng cơ bản của một hệ quản trị cơ sở dữ liệu, như nhập dữ liệu, tìm kiếm, thống kê.... (Xem bài toán quản lý mặt hàng dưới đây). Nói riêng, một số bài toán toán học đơn thuần cũng có thể cần tới tệp định kiểu. Những bài toán loại này ta không xét ở đây.

Bài toán quản lý mặt hàng:

Viết chương trình quản lý các mặt hàng theo các yêu cầu sau đây:

Thông tin về mỗi một mặt hàng bao gồm:

+ Mã hàng (là một dãy không quá 5 ký tự).
+ Loại hàng (là một trong các giá trị 1,2,3,4).
+ Số lượng (là một giá trị nguyên dương).
+ Đơn giá (là một giá trị thập phân, lấy chính xác đến phần trăm).

Chương trình phải tạo ra một menu gồm các chức năng:

1. Nhập thêm một mặt hàng mới vào cuối một file định kiểu có tên là MATHANG.DAT.
2. Tìm kiếm một mặt hàng theo mã hàng.
3. In ra màn hình danh sách các mặt hàng gồm mã hàng và giá trị tương ứng.
4. Tính tổng số lượng và tổng giá trị của một mặt hàng thuộc một loại nào đó.
5. Kết thúc chương trình.

http://tieulun.hopto.org

Bài làm:

```
Program QLHANG;
uses crt;
Const
   fname    =     'MATHANG.DAT';
type
  mathang=record
   mah   : string[5];
   loaih : byte;
   sl    : integer;
   dg    : real;
   end;
Var
  f        :     File of MATHANG;
  h        :     mathang;
  chon     :      byte;
Procedure Nhapmothang(Var h: Mathang);
Begin
 With h do
 Begin
  Write('ma hang: '); Readln(mah);
  Write('loai hang: '); Readln(loaih);
  Write('so luong: '); Readln(sl);
  Write('don gia: '); Readln(dg);
 End;
End;
Procedure Nhaphang;
Begin
 Nhapmothang(h);
 Seek(f,FileSize(f));
 Write(f,h);
 Writeln('Nhap xong...'); Readln;
End;
```


http://tieulun.hopto.org

```
Procedure TimHang;
Var Found: Boolean;
  ma: string[5];
Begin
 Write('Nhap ma hang can tim: '); Readln(ma);
 Found:= False;
 While Not Eof(f) and Not Found do
 Begin
  Read(f,h);
  If h.mah=ma then Found:= True;
 End;
 If Not Found Then Writeln('Khong co mat hang can tim')
 Else Writeln('Co mat hang can tim');
 Readln;
End;

Procedure DShang;
Begin
 While not Eof(f) do
 Begin
  Read(f,h);
  with h do Writeln(mah,' ',sl*dg:10:2);
 End;
 Readln;
End;

Procedure Thongke;
Var
   loai: byte;
   t: longint;
   s: real;
```


http://tieulun.hopto.org

```
Begin
 Write('Nhap loai hang can thong ke: '); Readln(loai);
 t:=0; s:=0;
 While not Eof(f) do
 Begin
  Read(f,h);
  with h do
  if loaih=loai then
  begin
    t:= t + sl;
    s:= s + sl*dg;
  end;
 End;
 Writeln('Tong so luong mat hang la ',t);
 Writeln('Tong gia tri cac mat hang la ',s:10:2);
 Readln;
End;
BEGIN
 Assign(f,fname);
 {$I-}
 Reset(f);
 {$I+}
 If IoResult>0 Then Rewrite(f);

 Repeat
  Clrscr;
  Seek(f,0);
  Writeln('1. Nhap mat hang moi');
  Writeln('2. Tim mat hang theo ma hang');
  Writeln('3. Danh sach mat hang');
  Writeln('4. Thong ke theo loai hang');
  Writeln('5. Thoat khoi chuong trinh');
```


http://tieulun.hopto.org

```
    Readln(chon);
    Case chon of
      1: Nhaphang;
      2: Timhang;
      3: Dshang;
      4: Thongke;
    End;
  Until chon=5;
   Close(f);
   Writeln('Tam biet!');
 END.
```

5. Câu hỏi và bài tập

Bài 5.6

Phân biệt tệp định kiểu và tệp văn bản về cách sử dụng?

Bài 5.7

So sánh tệp định kiểu với mảng một chiều?

Bài 5.8

Viết chương trình tạo tệp định kiểu, mỗi phần tử là một bản ghi biểu diễn thông tin của một học sinh gồm:

- Họ tên
- Năm sinh
- Lớp
- XL hạnh kiểm
- XL văn hoá

Chương trình làm nhiệm vụ quản lý một danh sách học sinh thông qua menu sau đây:

Menu quản lý học sinh:

1. Nhập học sinh mới.
2. Xem danh sách.
3. Tìm xem theo tên học sinh .
4. Thoát khỏi chương trình.

http://tieulun.hopto.org

Bài 5.9.

Viết chương trình quản lý sách trong một thư viện thông qua menu dưới đây. Biết rằng thông tin của một sách gồm:

- Tên sách
- Tác giả
- Năm xuất bản
- Nhà xuất bản

Menu quản lý sách:

1. Nhập các sách.
2. Xem danh mục sách.
3. Tìm sách theo tên tác giả.
4. Tìm sách theo tên gần đúng của sách.
5. Thoát khỏi chương trình.

Yêu cầu chương trình sử dụng tệp định kiểu, mỗi phần tử của tệp là một bản ghi mô tả thông tin cho một cuốn sách.

Giải thích chi tiết mục 4: Giả sử có 5 quyển sách:

1 - lap trinh pascal,

2 - huong dan lap trinh pascal,

3 - lap trinh C,

4 - huong dan lap trinh C,

5 - huong dan thiet ke website.

Khi đó:

Nếu nhập tên sách là	Kết quả tìm thấy các quyển sách
huong dan	2,4,5
pascal	1,2
C	3,4
huong dan C	4
thiet ke	5

http://tieulun.hopto.org

Phụ lục 1

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Thuật toán còn có một tính chất quan trọng đó là tính hiệu quả. Tuy nhiên việc đánh giá nghiêm túc hiệu quả của thuật toán liên quan đến sự đánh giá độ phức tạp của thuật toán. Chúng ta sẽ bước đầu tìm hiểu kỹ hơn về các vấn đề này.

I. TÍNH HIỆU QUẢ CỦA THUẬT TOÁN

Một thuật toán có tính hiệu quả, hiểu một cách đơn giản, là một thuật toán thực hiện nhanh, tốn ít thời gian.

Khi cài đặt thuật toán bằng một ngôn ngữ lập trình cụ thể, tính hiệu quả của thuật toán còn được xem xét trên góc độ về cấu trúc dữ liệu và dung lượng bộ nhớ của máy tính phải cấp phát để lưu trữ các dữ liệu mà thuật toán đòi hỏi. Tuy nhiên, hiệu suất thời gian của thuật toán thường được xem là quan trọng hơn.

Để đo được hiệu quả của thuật toán chúng ta cần tìm hiểu thêm về độ phức tạp của thuật toán.

Khi nói một thuật toán thực hiện nhanh, tốn ít thời gian, ta có thể hiểu là: so với thuật toán khác cùng giải quyết bài toán đang xét thì thuật toán của ta tốn ít phép tính hơn (tốn ít lệnh hơn).

Ví dụ 1.11: So sánh hiệu quả của 2 thuật toán dưới đây để tính giá trị hàm S(x,n):

$S(x,n) = 1 + x + x^2 + ... + x^n$ (n là số nguyên dương đủ lớn, x là số thực)

1. Thuật toán 1

Thuật toán 1 giả thiết sử dụng một hàm có sẵn LT(a,m) là hàm cho kết quả của a^m.

http://tieulun.hopto.org

Algorithm ChuoiSo.
Function Tính giá trị biểu thức $1 + x + x^2 + ... + x^n$
Input số nguyên $n > 0$, số thực x
Output $S(x,n) = 1 + x + x^2 + ... + x^n$
Format S(x,n)
Method
1. Khởi tạo S =1;
2. For i=1 to n làm việc sau đây
3. Thêm LT(x,i) vào S;
4. Return S;
End ChuoiSo.

2. Thuật toán 2

Algorithm ChuoiSo.
Function Tính giá trị biểu thức $1 + x + x^2 + ... + x^n$
Input số nguyên $n > 0$, số thực x.
Output $S(x,n) = 1 + x + x^2 + ... + x^n$
Format S(x,n)
Method
1. Khởi tạo S =1;
2. Khởi tạo T = x;
3. For i=1 to n làm các bước sau đây
4. a) Thêm T vào S;
5. b) Nhân T với x;
6. Return S;
End ChuoiSo.

Nhận xét: Hàm LT(a,m) khi được gọi ra để sử dụng sẽ thực hiện m phép nhân. Ta sẽ so sánh số phép nhân mà từng thuật toán trên phải thực hiện.

Trong thuật toán 1, câu lệnh số 3 và 4 thực hiện n lần gọi sử dụng hàm LT(x,i).

Lần 1 dùng 1 phép nhân.

Lần 2 dùng 2 phép nhân.

....

Lần n dùng n phép nhân.

http://tieulun.hopto.org

Vậy tổng số phép nhân là n*(n+1)/2.

Trong thuật toán 2, dễ nhận thấy các câu lệnh 3,4.5 yêu cầu máy tính thực hiện n phép nhân.

Như vậy thuật toán 2 hiệu quả hơn thuật toán 1 vì thuật toán 2 yêu cầu thực hiện ít phép nhân hơn, do đó thời gian thực hiện thuật toán 2 chắc chắn ít hơn thời gian thực hiện thuật toán 1.

II. ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

1. Khái niệm về độ tăng của hàm

Khái niệm về độ tăng của hàm đã được dùng trong toán học gần một thế kỷ nay. Trong tin học, độ tăng của hàm f(x) dùng để phân tích độ phức tạp thuật toán. Nhà toán học người Đức tên là Paul Bachmann là người đầu tiên đưa ra khái niệm về độ tăng của hàm năm 1892. Một nhà toán học người Đức khác tên là Edmund Landau đã đưa ra ký hiệu big-O (tức là ký pháp O lớn) để chỉ độ tăng của hàm. Sau đây là định nghĩa big-O (độ tăng của hàm):

Cho f(x) và g(x) là hai hàm số từ tập các số nguyên hoặc số thực đến tập các số thực. Ta nói rằng độ tăng của f(x) là hàm g(x) nếu tồn tại hai hằng số C và k sao cho:

$|f(x)| \leq C\,|g(x)|$ *với mọi* $x > k$. (*)

Về mặt ký hiệu: độ tăng của f(x) là g(x) được ký hiệu là:

$f(x) = O(g(x))$

Trong tin học, khi phân tích độ phức tạp thuật toán, nếu ta tìm được $f(x) = O(g(x))$ *ta thường nói rằng "f(x) có bậc g(x) " và ta sẽ sử dụng cách gọi này.* Ngoài ra, trong tin học, các hàm được xem xét chỉ nhận giá trị dương nên khi không nhầm lẫn, ta sẽ bỏ dấu giá trị tuyệt đối trong quan hệ (*) ở trên.

Ví dụ 1:

Chứng minh rằng $f(x) = x^2 + 2x + 1$ có bậc $g(x) = x^2$.

Giải:

Vì $0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$ với mọi $x > 1$, từ đó, theo định nghĩa big-O ta chọn $C = 4$, $k=1$ và $g(x) = x^2$ thì ta được $f(x) = O(x^2)$ (điều phải chứng minh).

http://tieulun.hopto.org

Chú ý: Theo chứng minh trên ta có thể chọn $g(x)= 4x^2$ khi đó ta sẽ chọn C=1 và k = 1. Nhưng về mặt quy ước, khi dùng khái niệm big-O, hàm g(x) trong quan hệ f(x) là O(g(x)) được chọn phải là hàm nhỏ nhất có thể được và tốt nhất là các hàm sơ cấp, ví dụ như x, x^2, $\log_2 x$,...

Ví dụ 2:

Chứng minh rằng $f(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ với a_i là các số thực (i=0,1,..., n) có bậc $g(x) = O(x^n)$.

Giải:

Dùng bất đẳng thức tam giác, nếu x > 1 ta có:

$$\begin{aligned} |g(x)| &= |a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0| \\ &\le |a_n|x^n + |a_{n-1}|x^{n-1} + ... + |a_1|x + |a_0| \\ &= x^n (|a_n| + |a_{n-1}| / x + ... + |a_1| / x^{n-1} + |a_0| / x^n) \\ &\le x^n (|a_n| + |a_{n-1}| + ... + |a_1| + |a_0|) \end{aligned}$$

Điều này chứng tỏ rằng:

$|f(x)| \le C\, x^n$ với mọi x > k=1

Ở đây $C = |a_n| + |a_{n-1}| + ... + |a_1| + |a_0|$. Từ đó, theo định nghĩa big-O ta suy ra $f(x) = O(x^n)$ (điều phải chứng minh).

2. Độ phức tạp thuật toán

Độ phức tạp của một thuật toán sẽ đánh giá một cách cụ thể hiệu quả của thuật toán đó.

Sự phân tích bộ nhớ của máy tính liên quan đến độ phức tạp không gian của thuật toán. Sự phân tích thời gian cần thiết để máy tính giải một bài toán liên quan đến độ phức tạp thời gian của thuật toán.

Độ phức tạp không gian của thuật toán liên quan đến cấu trúc dữ liệu của ngôn ngữ lập trình đang được sử dụng và cách cài đặt cấu trúc dữ liệu đó. Chúng ta không xem xét đến cấu trúc dữ liệu ở đây, vì vậy, việc đánh giá thuật toán (độ phức tạp thuật toán) được xem xét về độ phức tạp thời gian.

Thời gian thực hiện một thuật toán chịu ảnh hưởng của nhiều yếu tố. Đương nhiên một yếu tố trong đó là kích thước dữ liệu vào. Ví dụ, trong thuật toán tính S là tổng của các bình phương của n số tự nhiên đầu tiên thì thời gian cần thiết để cộng dần các số hạng i^2 (i =1,2,..., n) vào tổng S chắc chắn phụ thuộc vào giá trị đầu vào n. Như vậy, *thời gian thực hiện T của một thuật toán phải được biểu diễn như là hàm T(n) của độ lớn dữ liệu vào n.*

http://tieulun.hopto.org

Các kiểu lệnh (instruction) và vận tốc của máy tính thực hiện những lệnh đó cũng ảnh hưởng đến thời gian thực hiện. tuy nhiên, những yếu tố này phụ thuộc vào máy tính đang được sử dụng, vì vậy chúng ta không thể biểu diễn giá trị của T(n) một cách đầy đủ bằng các đơn vị thời gian, chẳng hạn như giây. Thay vào đó, *T(n) sẽ được tính gần đúng như là số các lệnh (instruction) được thực hiện.*

Một yếu tố khác có ảnh hưởng đến thời gian tính toán là chất lượng của chương trình do bộ dịch tạo ra. Không phải tất các các bộ dịch đều tạo ra những chương trình có hiệu suất như nhau, từ đó suy ra rằng T(n) không thể được tính như là số các chỉ thị máy (machine instructions). *Vì vậy, T(n) sẽ được tính như là số lần thực hiện các lệnh trong thuật toán (hay trong chương trình cài đặt thuật toán đó).*

Bây giờ ta sẽ tính hàm T(n) cho một số thuật toán.

Độ phức tạp của một thuật toán chính là bậc (hay độ tăng) của hàm T(n).

Ví dụ 1.12: Thuật toán tính giá trị trung bình

Algorithm Average.

Function Nhập n số và tính giá trị trung bình của chúng .

Input n > 0 và n số

Output Mean = giá trị trung bình của n số được nhập.

Format Mean = Average(a_1,a_2,...,a_n)

Method

1. Nhập n;
2. Khởi tạo S =0;
3. Khởi tạo i =0;
4. While i <= n làm các bước sau đây:
5. a) Đọc a;
6. b) Thêm a vào S;
7. c) Tăng i thêm 1;
8. Tính Mean = S/n;

End Average.

http://tieulun.hopto.org

Các lệnh 1, 2 và 3 được thực hiện một lần. Các lệnh 5, 6 và 7 tạo ra thân của vòng lặp được thực hiện mỗi lệnh n lần, và lệnh 4 kiểm tra sự lặp lại, được thực hiện n+1 lần, bởi phải thêm một lần kiểm tra bổ sung để xem biến i đã vượt qua giá trị n hay chưa. Sau khi kết thúc vòng lặp, lệnh 8 được thực hiện một lần.

Lệnh	Số lần thực hiện
1	1
2	1
3	1
4	n +1
5	n
6	n
7	n
8	1
Tổng cộng	4n + 5

Thời gian thực hiện thuật toán này là:

$T(n) = 4n + 5$

Ta có:

$4n+5 \leq 4n + n = 5n$ với mọi $n >= 5$.

Vậy theo định nghĩa big-O, chọn $C = 5$, $k = 1$, $g(n) = n$ ta có:

$T(n) = O(n)$.

Ta nói rằng thuật toán tính giá trị trung bình của n số có độ phức tạp bậc n.

Nhận xét: Thời gian tính toán như ở ví dụ 1.12 chỉ phụ thuộc vào kích thước dữ liệu đầu vào. Trong nhiều vấn đề khác, thời gian tính toán còn phụ thuộc vào thứ tự các mục của dữ liệu đầu vào. Ví dụ thời gian sắp xếp một dãy các số nguyên đã được sắp xếp gần hết sẽ nhanh hơn thời gian sắp xếp một dãy các số nguyên được sắp xếp theo thứ tự ngược lại. Chúng ta có thể đo được T trong trường hợp ***tồi nhất***, ***tốt nhất***, hoặc chúng ta có thể tính được giá trị

http://tieulun.hopto.org

trung bình của T trong tất cả các trường hợp có thể có. Cách theo trường hợp tốt nhất thường không cung cấp nhiều thông tin lắm, còn cách tính giá trị trung bình thường khó xác định hơn là cách theo trường hợp xấu nhất. Vì vậy, T(n) thường được tính như là sự thực hiện thuật toán trong trường hợp xấu nhất của dữ liệu vào. Ta xét ví dụ sau đây:

Ví dụ 1.13: Thuật toán sắp xếp thứ tự kiểu chọn lựa (Selection Sort)

Algorithm Selection_Sort.

Function Sắp xếp n số $a_1,a_2,....,a_n$ theo thứ tự tăng dần.

Input n > 0 và n số $a_1,a_2,...,a_n$

Output $a_1,a_2,...,a_n$ được sắp xếp tăng dần.

Format Sort($a_1,a_2,...,a_n$)

Method

1. For i=1 to n-1 làm các bước sau:

(* Chọn phần tử nhỏ nhất trong dãy a_i , .. , a_n *)

2. a. Gán giá trị i cho k;
3. b. Gán giá trị a_i cho min;
4. c. For j = i +1 to n làm các bước sau:
5. If a_j < min (*tìm ra phần tử nhỏ hơn*) Then
6. i. Gán giá trị j vào k;
7. ii. Gán giá trị a_j vào min;
8. d. If i<>k (*phần tử nhỏ nhất không ở đầu dãy) Then

(*Chuyển phần tử nhỏ nhất này về đầu dãy*)

9. i. Gán giá trị a_i vào a_k ;
10. ii. Gán giá trị min vào a_i ;

End Selection_Sort.

Trường hợp xấu nhất: Ban đầu dãy $a_1,a_2,...,a_n$ "bị" sắp xếp giảm dần. Khi đó, lệnh 5 thực hiện bao nhiêu lần thì các lệnh 6 và 7 thực hiện bấy nhiêu lần (do điều kiện sau if của 5 luôn đúng). Tương tự như thế, số lần thực hiện các lệnh 9 và 10 bằng số lần thực hiện lệnh 8.

Lệnh 1 được thực hiện n lần (i chạy từ 1 đến n, giá trị i = n làm kết thúc vòng lặp). Các lệnh 2, 3, 8 (do đó cả 9 và 10) thực hiện n - 1 lần trong vòng lặp

http://tieulun.hopto.org

for i. Khi đi vào vòng lặp for i lần thứ nhất: lệnh 4 thực hiện n lần, lệnh 5 (do đó cả 6 và 7) thực hiện n - 1 lần. Khi đi vào vòng lặp for i lần thứ hai: lệnh 4 thực hiện n - 1 lần, các lệnh 5, 6, 7 thực hiện n - 2 lần,... Như vậy lệnh 4 thực hiện tổng cộng là n + (n - 1) + .. + 2 = (n(n + 1)/2) - 1 lần, các lệnh 5, 6, 7 mỗi lệnh thực hiện (n - 1) + (n - 2) + ... + 1 = n(n - 1)/2 lần. Vậy thời gian tổng cộng của thuật toán trên là:

$$T(n) = n + 5(n - 1) + (n(n + 1)/2) - 1 + 3(n(n - 1)/2)$$
$$= 2n^2 + 4n - 5$$

Bởi vì $n \le n^2$ với mọi $n >= 0$ nên ta có:

$2n^2 + 4n - 5 \le 2n^2 + 4n^2 = 6n^2$ với mọi $n >= 0$

Vậy theo định nghĩa big-O, chọn $C = 6$, $k = 0$, ta có:

$T(n) = O(n^2)$

Vậy thuật toán sắp xếp kiểu chọn trực tiếp có độ phức tạp bậc n^2.

Nhận xét: Nếu hai thuật toán cùng giải quyết một bài toán có độ phức tạp khác nhau, thuật toán có bậc thấp hơn thường được ưu tiên hơn. Ví dụ nếu thuật toán 1 có thời gian tính $T_1(n) = O(n)$, thuật toán 2 có thời gian tính $T_2(n) = O(n^2)$ thì thuật toán 1 được xem là tốt hơn thuật toán 2, bởi vì nó sẽ thực hiện hiệu quả hơn đối với giá trị lớn của dữ liệu vào n. Tuy nhiên khi dữ liệu vào n đủ nhỏ, có thể thuật toán có bậc nhỏ hơn sẽ kém hiệu quả hơn thuật toán có bậc cao hơn. Ví dụ $T_1(n) = 10n$ còn $T_2 = 0.1n^2$. Vì $10n > 0.1n^2$ với $n < 100$ nên trong trường hợp này, thuật toán ứng với O(n) lại không hiệu quả bằng thuật toán $O(n^2)$. Nhưng khi $n > 100$ thì thuật toán với độ phức tạp O(n) rõ ràng lại hiệu quả hơn thuật toán có độ phức tạp $O(n^2)$, và, khi n đủ lớn thì tính hiệu quả của thuật toán 1 càng rõ rệt. Chúng ta xem xét tiếp hai ví dụ sau đây:

Ví dụ 1.14: Thuật toán tìm kiếm tuyến tính (Linear searching)

Algorithm Linear_Searching.
Function Tìm phần tử x trong dãy $a_1,a_2,...,a_n$
Input Số nguyên n, số thực x, dãy n số thực $a_1,a_2,...,a_n$
Output Biến boolean Found và biến nguyên k (Found = True nếu tìm thấy x. tại vị trí a_k, Found = False nếu không tìm thấy x).
Format Linear_Searching($a_1,a_2,...,a_n$, x, Found, k);
Method

http://tieulun.hopto.org

1. Khởi gán Found = False;
2. Khởi gán k = 1;
3. While (k <= n) and Not Found làm các bước sau:
4. If a_k = x then
5. Ghi nhận Found = True
6. Else Tăng k thêm 1;

End Linear_Searching.

Trường hợp xấu nhất là trường hợp không có x trong dãy (a_n) đã cho. Thời gian tính $T_L(n)$ cho thuật toán tìm kiếm tuyến tính là như sau:

Lệnh	Số lần thực hiện
1	1
2	1
3	n +1
4	n
5	0
6	n
Tổng cộng	3n + 3

Vậy $T_L(n) = 3n + n$ hay $T_L(n) = O(n)$.

bởi vì $3n+3 \leq 4n$ với mọi $n \leq 3$.

Nếu dãy (a_n) đã được sắp xếp, giả sử đó là sắp xếp tăng dần, thì thuật toán tìm kiếm nhị phân sau đây có thể sử dụng thay cho phép tìm kiếm tuyến tính.

Ví dụ 1.15: Thuật toán tìm kiếm nhị phân (Binary searching)

Algorithm Binary_Searching.

Function Tìm phần tử x trong dãy $a_1,a_2,...,a_n$ đã được sắp.

Input Số nguyên n, số thực x, dãy n số thực $a_1,a_2,...,a_n$ đã được sắp xếp theo thứ tự tăng dần.

Output Biến boolean Found và biến nguyên k (Found = True nếu tìm thấy x tại vị trí a_k , Found = False nếu không tìm thấy x).

Format Binary_Searching($a_1,a_2,...,a_n$, x, Found, k);

http://tieulun.hopto.org

Method

1. Khởi gán Found = False;
2. Khởi gán Left = 1;
3. Khởi gán Right = n;
4. While (Left <= Right) and Not Found làm các bước sau:
5. a. Tính k = (Left + Right) div 2;

(* div là phép chia lấy phần nguyên).

6. b. If $x < a_k$ Then
7. Gán Right = k-1
8. Else If $x > a_k$ Then
9. Gán Left = k +1
10. Else

Ghi nhận Found = Truc

End Binary_Searching.

Trong thuật toán này, các lệnh 1, 2, 3 thực hiện đúng một lần. Để tính được độ phức tạp thuật toán $T_B(n)$ trong trường hợp tồi nhất, ta phải xét số lần thực hiện của vòng lặp chứa các lệnh từ 4 đến 10.

Mỗi lần đi qua vòng lặp này, độ lớn của dãy con được giảm đi để tìm trong một nửa dãy con (bên trái hoặc bên phải dãy con trong lần tìm trước đó). Lần cuối cùng đi qua vòng lặp là khi dãy con chỉ còn một phần tử.

Như vậy, số lần lặp lại của vòng lặp này là 1 cộng với k lần đi qua vòng lặp để cuối cùng chỉ duyệt trên dãy con có độ dài 1.

Ta thấy độ lớn của dãy con sau k lần đi qua nhiều nhất là $n/2^k$, ta phải có: $n/2^k < 2$, nghĩa là $n < 2^{k+1}$, hay:

$\log_2 n < k + 1$ (*)

Số lần đi qua vòng lặp là số nguyên bé nhất thoả mãn bất đẳng thức trên, nghĩa là phần nguyên của $\log_2 n$. Như vậy, trong trường hợp tồi nhất, khi x lớn hơn các phần tử $a_1, a_2, ..., a_n$, lệnh 4 được thực hiện không nhiều hơn $2 + \log_2 n$ lần, lệnh 5, 6 và 8 không nhiều hơn $1 + \log_2 n$ lần, các lệnh 7, 9, 10 thực hiện 0 lần. Thời gian tính tổng cộng như vậy là không quá $8 + 4\log_2 n$, tức là:

$T_B(n) = O(\log_2 n)$

Nhận xét: Xét 2 thuật toán: thuật toán tìm kiếm tuyến tính có độ phức tạp O(n) và thuật toán tìm kiếm nhị phân có độ phức tạp $O(\log_2 n)$.

http://tieulun.hopto.org

Thấy rằng phép tìm kiếm nhị phân hiệu quả hơn phép tìm kiếm tuyến tính đối với các dãy lớn. Tuy nhiên đối với các dãy nhỏ (thực nghiệm với $n \leq 20$ phần tử) thì ta thấy phép tìm kiếm tuyến tính lại tốt hơn phép tìm kiếm nhị phân.

Độ phức tạp của các thuật toán thường rơi vào các trường hợp sau đây:

Độ phức tạp thuật toán	Thuật ngữ
O(1)	Độ phức tạp hằng số.
$O(\log_2 n)$	Độ phức tạp logarit.
O(n)	Độ phức tạp tuyến tính.
$O(n\log_2 n)$	Độ phức tạp $n\log_2 n$.
$O(n^k)$	Độ phức tạp đa thức.
$O(a^n)$	Độ phức tạp hàm mũ.
O(n!)	Độ phức tạp giai thừa.

Bảng dưới đây là một ví dụ để tham khảo thời gian cần thiết để thực hiện T(n) lệnh đối với các hàm thông dụng tính độ phức tạp với n = 256, giả sử mỗi lệnh thực hiện trong 1 micro giây:

Hàm	Thời gian
$\log \log_2 n$	3 micro giây
$\log_2 n$	8 micro giây
n	0.25 mili giây
$n\log_2 n$	2 mili giây
n^2	65 mili giây
n^3	17 giây
2^n	3.7×10^{61} thế kỷ

3. Câu hỏi và bài tập

Bài PL1.1

Hãy đưa ra một ví dụ của thuật toán có độ phức tạp O(1).

Bài PL1.2

Hãy cho một ví dụ để giải thích tại sao nếu $T(n) = O(n)$ thì cũng đúng khi ta viết $T(n) = O(n^2)$.

http://tieulun.hopto.org

Bài PL1.3

Hãy tính độ phức tạp tính toán của thuật toán tính a^n.

Bài PL1.4

Hãy tính độ phức tạp tính toán của thuật toán tính ước số chung lớn nhất của 2 số nguyên a và b.

http://tieulun.hopto.org

Phụ lục 2

THUẬT TOÁN ĐỆ QUY VÀ CHƯƠNG TRÌNH CON ĐỆ QUY

Chúng ta đã tìm hiểu về khái niệm thuật toán, các tính chất của thuật toán, các phương pháp diễn tả thuật toán và tiến thêm một bước nữa là đánh giá độ phức tạp của thuật toán. Thuật toán, đúng như định nghĩa của nó, cuối cùng phải đạt đến mục tiêu là tính khả thi trong việc thông qua một ngôn ngữ lập trình cụ thể để chỉ dẫn cho máy tính cách giải bài toán đề ra. Các ví dụ và bài tập trong phần kiến thức trọng tâm như: Các cấu trúc điều khiển, mảng, xâu ký tự đã là những minh hoạ cụ thể về việc cài đặt thuật toán bằng lập trình Pascal.

Nếu muốn nâng cao kiến thức, chúng ta còn phải đi tiếp một chặng đường khá dài mới có thể tiếp cận tương đối đầy đủ các kiến thức về thuật toán. Nếu có điều kiện, ta có thể tìm hiểu thêm về: Thuật toán đệ quy, thuật toán duyệt bằng đệ quy (Back tracking), các thuật toán tìm kiếm và sắp xếp nâng cao, các thuật toán đồ thị,... và nói chung, khi đó chúng ta phải tham khảo thêm các kiến thức về toán học rời rạc và những lĩnh vực liên quan khác. Trong phần kiến thức đệ quy này, ta sẽ tìm hiểu về thuật toán đệ quy và chương trình con đệ quy.

I. CÁC KHÁI NIỆM VỀ ĐỆ QUY

Khái niệm đối tượng đệ quy dẫn đến một loạt các khái niệm như: bài toán đệ quy, lời giải đệ quy, thuật toán đệ quy và cuối cùng là chương trình con đệ quy.

Ta nói: Một đối tượng là đệ quy nếu nó bao gồm chính nó như một bộ phận hoặc nó được định nghĩa dưới dạng của chính nó.

http://tieulun.hopto.org

Bài toán T là bài toán đệ quy nếu nó được giải bằng một bài toán T' có dạng giống như T, nói cách khác T là bài toán được giải bằng một thuật toán đệ quy.

Ví dụ về hình ảnh đệ quy: Một cái túi đựng trong một cái túi thứ 2, mà cái túi thứ 2 này là một cái túi đựng trong một cái túi thứ 3, mà cái túi thứ 3 này là một cái túi đựng trong một cái túi thứ 4... Tuy nhiên quá trình các túi chứa trong nhau ấy không vô hạn, đến một cái túi thứ n (n xác định) thì không đựng trong cái túi nào nữa.

Trong toán học, ta gặp rất nhiều định nghĩa đệ quy mà thường là các công thức để tính giá trị cho một hàm nào đó có thể tính được nhờ quy nạp toán học:

Ví dụ 1: Tính f(n) = n!

f(n) = 1 nếu n = 0

f(n) = f(n - 1)*n nếu n > 0 (vì n! = (n - 1)!*n)

Có thể coi bài toán T là bài toán tính f(n), được giải bằng bài toán T' là f(n - 1) có dạng giống bài toán T. Bài toán T' lại được giải bằng bài toán T'' có dạng giống như T'. Tức là f(n - 1) = f(n - 2)*(n - 1). Cứ tiếp tục quá trình đó đến khi thu được bài toán T^{n} tính f(0) thì được giải bằng một phương pháp riêng đã biết, ở đây f(0) = 1, theo quy ước.

Xét dãy quá trình giải: $T \leftarrow T' \leftarrow T'' \leftarrow ... \leftarrow T^{n} = T_0$.

(Ký hiệu ← gọi là "được giải bằng").

Quá trình giải bài toán T nói trên chia thành 2 phần:

- Phần đệ quy là quá trình: $T \leftarrow T' \leftarrow T'' \leftarrow ... \leftarrow T^{n}$
- Phần "neo" là việc giải bài toán $T^{n} = T_0$ bằng một phương pháp khác hẳn, đã biết.

Như vậy quá trình thực hiện một lời giải đệ quy (hay thuật toán đệ quy) là một quá trình luôn dừng lại sau một số hữu hạn bước. Một định nghĩa đệ quy mà không có phần neo sẽ làm vi phạm tính dừng của thuật toán đệ quy tương ứng với nó.

Số lượng các bài toán trong phần đệ quy xác định là độ sâu hay "mức" của đệ quy. Ví dụ bài toán tính f(n) = n! có độ sâu là n.

Ví dụ 2: Tính F(n) = Số hạng thứ n của dãy số Fibonaci

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,...:

http://tieulun.hopto.org

F(n) = 1 nếu n = 1 hoặc n = 2. (Phần neo)

F(n) = F(n - 1) + F(n - 2) nếu n >= 3. (Phần đệ quy)

Ví dụ 3: Tính U(a,b) là hàm cho biết ước số chung lớn nhất của hai số nguyên dương a và b cho trước.

Cách 1: Theo thuật toán trừ liên tiếp:

U(a,b) = b nếu a = b (phần neo).

U(a,b) = U(a - b, b) nếu a > b.

U(a,b) = U(a,b - a) nếu a < b (phần đệ quy).

Cách 2: Theo thuật toán chia liên tiếp:

U(a,b) = b nếu a mod b = 0 (phần neo).

U(a,b) =(b, a mod b) nếu a mod b >0 (phần đệ quy).

II. CHƯƠNG TRÌNH CON ĐỆ QUY

1. Cách viết một chương trình con đệ quy

Chương trình con thể hiện một thuật toán đệ quy gọi là chương trình con đệ quy.

Định nghĩa một chương trình con đệ quy gồm 2 phần:

1. Phần neo: Lời gọi hàm hay thủ tục được thực hiện bằng một lời giải riêng đã biết.

2. Phần đệ quy: Lời gọi chính hàm hay thủ tục đó nhưng có kích thước dữ liệu đầu vào thay đổi, theo xu hướng (thường là nhỏ hơn) để quá trình đệ quy dẫn đến phần neo.

Ví dụ 1: Hàm tính n!

```
Function f(n:integer):longint;
Begin
If n=0 then f:=1
Else f:=f(n-1)*n;
End;
```

Ví dụ 2: Hàm tính số hạng Fibonaci thứ n.

```
Function g(n:integer):longint;
Begin
```


http://tieulun.hopto.org

```
If (n=1)or(n=2) then g:=1
Else g:=g(n-1) + g(n-2);
End;
```

Ví dụ 3: Hàm tính USCLN của 2 số nguyên dương a và b.

```
Function U(a,b:integer):integer;
Begin If (a=b) then U:=b
Else U:=U(b, a mod b);
End;
```

2. Bản chất của chương trình con đệ quy

2.1. Nhận xét

Xét hàm đệ quy tính n!.

Chẳng hạn, cần tính f(4). Ta thấy rằng hàm f với n = 4 được gọi trước nhưng lại không cho ngay giá trị của hàm với chính n = 4 mà tiếp tục phải gọi hàm với n = 3. Tương tự, mặc dù hàm f với n = 3 được gọi trước nhưng giá trị của nó cũng chưa được xác định ngay với chính n = 3 mà tiếp tục lại gọi hàm với n = 2,...

Một cách tổng quát: Hàm ở mức k chưa hoàn tất giá trị tính toán nếu hàm ở mức k + 1 chưa được tính toán xong mặc dù mức k được triệu gọi trước.

Vấn đề đặt ra là cơ chế nào để sau khi đến phần neo, tính được f(0) = 1, chương trình sẽ quay lại lời gọi f(1) = f(0)*1 để hoàn tất tính f(1) = 1, tương tự như thế, quay lại lời gọi f(2) = f(1)*2 để hoàn tất tính f(2) = 2,..., cứ tiếp tục như vậy cho đến khi quay lại lời gọi f(4) = f(3)*4 = 6*4 và hoàn tất tính f(4) = 24.

Một cách chính xác, phải có một cơ chế lưu ngữ cảnh trong lời gọi hàm ở mức k trước khi triệu gọi chính hàm đó ở mức k + 1, để "sau này" mới có thể quay lại và khôi phục ngữ cảnh (câu lệnh và giá trị các biến) đã được lưu để hoàn tất lời gọi hàm ở mức k trước đó.

2.2. Bản chất của chương trình con đệ quy

Dưới góc độ lập trình, chương trình chia bộ nhớ thành ba loại:

- Bộ nhớ quy ước (coventionnal memory).
- Bộ nhớ ngăn xếp (stack memory).

http://tieulun.hopto.org

• Bộ nhớ động (heap blocks memory).

Bộ nhớ quy ước (thường không quá 64KB) để cấp phát cho các biến toàn cục trong chương trình. Bộ nhớ heap để cấp phát cho các biến động (là các biến khi chạy chương trình, nếu có nhu cầu, mới đòi hỏi cấp phát bộ nhớ), dung lượng của heap có thể khá nhiều. Bộ nhớ stack (thường không vượt quá 16KB) để cấp phát cho các biến địa phương và đặc biệt để chi phí cho chương trình con đệ quy.

Bản chất của quá trình đệ quy thể hiện ở việc dùng stack để lưu ngữ cảnh lời gọi chương trình con trong quá trình đệ quy. Trước khi một chương trình con đệ quy từ mức k sâu vào mức k + 1 thì ngữ cảnh lời gọi nó được lưu vào ngăn xếp. Ngữ cảnh lời gọi chương trình con đệ quy tại một mức k nào đó bao gồm: lệnh gọi chương trình con và tất cả các giá trị của các biến liên quan tại mức k này. Điều này giải thích tại sao trong chương trình con đệ quy chỉ có một số rất ít biến nhưng các biến đó lại có thể chứa vô số giá trị, các giá trị không bị đè lên nhau mà được ghi nhớ lại và được gọi sử dụng lại. Như vậy, sau khi quá trình đệ quy gặp phần neo thì các ngữ cảnh lời gọi chương trình con được lưu trước đó lần lượt được lấy dần ra để "khôi phục" lại lời gọi cũ và hoàn tất việc tính toán tại mức tương ứng.

Công việc sử dụng stack để lưu ngữ cảnh trong quá trình đệ quy và dần dần khôi phục các ngữ cảnh được chương trình dịch đảm nhận và người lập trình không cần chủ động làm việc đó. Tuy nhiên người lập trình lại rất cần thiết hiểu được "cơ chế" đó thì mới có thể "viết tốt" các chương trình con đệ quy. Từ "viết tốt" được hiểu là: lượng hóa được độ sâu của đệ quy để tránh không tràn stack và có thể chủ động quản lý gán giá trị và tự khôi phục giá trị cho các biến toàn cục được sử dụng trong chương trình con đệ quy.

2.3. Ba cách nhìn nhận

Bây giờ hàm tính f(n) = n! có 3 cách xây dựng:

Cách 1: Dùng phương pháp lặp.

Cách 2: Dùng phương pháp đệ quy, trong đó, tự chương trình dịch quản lý stack.

Cách 3: Dùng phương pháp mô phỏng đệ quy, trong đó, tự người lập trình tạo ra stack để thể hiện quá trình đệ quy.

http://tieulun.hopto.org

Cụ thể như sau:

Cách 1: Dùng phương pháp lặp

```
Function f(n:integer):longint;
var i: integer; T:longint;
Begin
T:=1;
For i:=1 to n do T:=T*i;
f:=T;
End;
```

Cách 2: Dùng phương pháp đệ quy

```
Function g(n:integer):longint;
Begin
If n=0 then g:=1
Else g:=g(n-1)*n;
End;
```

Cách 3: Dùng phương pháp mô phỏng đệ quy

```
Function h(n:integer):longint;
Var
m:integer;
Stack[1..100] of integer;
top:integer;
Begin
m:=n; top:=0;
while n>0 do
begin
top:=top+1;
Stack[top]:=n;
n:=n-1;
end;
If n=0 then
```


http://tieulun.hopto.org

```
begin
kq:=1;
while top>0 do
begin
kq:=kq*Stack[top];
top:=top-1;
end;
h:=kq;
end;

End;
```

Thoạt đầu ta cho rằng sự hiểu biết cách thứ 3 là không cần thiết vì thực tế nó không tiết kiệm thời gian hơn một chương trình con đệ quy thực sự. Nhưng hiểu về nó có hai lợi ích: Thứ nhất, ta có thể quản lý được bộ nhớ thay vì ngăn xếp nên ta có thể tăng cường bộ nhớ cho quá trình đệ quy trong trường hợp chương trình dịch không đảm đương được việc đó, dẫn hướng biên dịch về gia tăng bộ nhớ stack cũng không làm được điều đó và sự cài đặt chi tiết dữ liệu kết hợp với các ý đồ chiến thuật trong thuật toán mới có thể được thể hiện trong cách này; Thứ hai, việc nắm được bản chất quá trình đệ quy như vậy giúp ta hiểu được những chương trình con đệ quy trừu tượng hơn mà ta sẽ xem xét dưới đây.

III. PHÂN TÍCH QUY NẠP TRONG ĐỆ QUY

Không phải bài toán đệ quy nào cũng dễ dàng viết được chương trình con đệ quy như các ví dụ đã nêu. Nói chung, những bài toán mà tìm được ngữ nghĩa toán học để định nghĩa nó một cách đệ quy thì đều có thể từ đó viết được chương trình con đệ quy để giải nó một cách thuận lợi hơn.

Vậy những bài toán đệ quy gọi là "khó" hiểu theo nghĩa là "khó biểu diễn lời giải đệ quy cho nó bằng quy nạp toán học hình thức" (rút cuộc đó chính là thuật toán).

Dưới đây là một số bài toán đệ quy tiêu biểu và những gợi ý để tiếp cận một thuật giải đệ quy cho bài toán đó bằng phương pháp quy nạp toán học.

http://tieulun.hopto.org

1. Bài toán Tháp Hà Nội

Có 3 cọc A, B, C và có n đĩa tròn đục lỗ ở giữa, có bán kính phân biệt. Ban đầu n đĩa được xếp trên cọc A bán kính tăng dần từ trên xuống dưới, nghĩa là đĩa bé nhất đặt lên trên cùng, đĩa to hơn nằm phía dưới và cuối cùng là đĩa to nhất. Cần viết chương trình thể hiện các bước chuyển n đĩa từ cọc A sang cọc C, lấy cọc B làm trung gian sao cho:

- Mỗi bước chỉ được chuyển một đĩa.
- Khi xếp các đĩa vào một cọc nào đó phải đảm bảo với 2 đĩa xếp cạnh nhau bất kỳ thì đĩa bé nằm lên trên đĩa to (n đủ nhỏ).

1.1. Quy nạp toán học

Ta gọi T là bài toán thể hiện cách chuyển n đĩa từ cọc A sang cọc C lấy cọc B làm cọc trung gian.

- Trường hợp n = 1: Bài toán giải đơn giản, ta chỉ việc chuyển 1 đĩa (đó) từ cọc A sang cọc C. Vậy bài toán T với n = 1 là giải được.

- Trường hợp n = 2: Bài toán T giải được vì nó được phân ra thành 3 bài toán đã giải được với n = 1:

Chuyển 1 đĩa từ cọc A sang cọc B, cọc C làm trung gian.

Chuyển 1 đĩa từ cọc A sang cọc C, cọc B làm trung gian.

Chuyển 1 đĩa từ cọc B sang cọc C, cọc A làm trung gian.

Vậy bài toán T với n = 2 là giải được.

Lưu ý, việc ghi tên các cọc trung gian trong các bước chuyển trên trong trường hợp n = 2 là thừa vì ta không sử dụng đến các cọc trung gian này. Nhưng để cho đầy đủ ta vẫn chỉ ra các cọc trung gian đó. Hơn nữa, nếu nhìn tổng thể thì việc chuyển 2 đĩa từ cọc A sang cọc C đã sử dụng tới cọc trung gian B.

- Trường hợp n = 3: Bài toán T cũng giải được vì nó được phân ra thành 3 bài toán đã giải được với n = 2 và n = 1:

Chuyển 2 đĩa từ cọc A sang cọc B, cọc C làm trung gian.

Chuyển 1 đĩa từ cọc A sang cọc C, cọc B làm trung gian.

Chuyển 2 đĩa từ cọc B sang cọc C, cọc A làm trung gian.

- Tổng quát, nếu bài toán T đã giải được với số đĩa là 1,2,...,n - 1 thì sẽ giải được với số đĩa là n. Vì bài toán T với n đĩa chia thành thành 3 bài toán đã giải được với số đĩa là n - 1 và 1:

http://tieulun.hopto.org

Chuyển n - 1 đĩa từ cọc A sang cọc B, cọc C làm trung gian

Chuyển 1 đĩa từ cọc A sang cọc C, cọc B làm trung gian.

Chuyển n - 1 đĩa từ cọc B sang cọc C, cọc A làm trung gian.

1.2. Phân tích thủ tục đệ quy

Ta gọi thủ tục đệ quy là Chuyển(n,c1,c3,c2) để thể hiện bài toán T(n), tức là ***mô tả cách chuyển n đĩa từ cọc c1 sang cọc c3, lấy cọc c2 làm trung gian.*** (Chú ý rằng c1,c2,c3 là các biến chứa tên các cọc).

- Trường hợp n = 1, T(1) là Chuyển(1,c1,c3,c2).

Đây chính là phần neo của thủ tục đệ quy. Ta chỉ việc in ra việc chuyển 1 đĩa từ cọc c1 sang cọc c3.

- Trường hợp n = 2, bài toán T(2) là Chuyển(2,c1,c3,c2) dựa vào bài toán T(1) đã giải:

Chuyển(1,c1,c2,c3); (1)

Chuyển(1,c1,c3,c2);

Chuyển(1,c2,c3,c1);

- Trường hợp n = 3, bài toán T(3) là Chuyển(3,c1,c3,c2) dựa vào 2 bài toán T(1) và T(2) đã giải:

Chuyển(2,c1,c2,c3);

Chuyển(1,c1,c3,c2);

Chuyển(2,c2,c3,c1);

- Trường hợp tổng quát, bài toán T(n) là Chuyển(n,c1,c3,c2) dựa vào 2 bài toán T(1) và T(n - 1) đã giải:

Chuyển(n - 1,c1,c2,c3);

Chuyển(1,c1,c3,c2);

Chuyển(n - 1,c2,c3,c1);

1.3. Thủ tục đệ quy

```
Procedure Chuyen(n:byte; c1,c3,c2: char);
Begin
If n = 1 then Writeln('Chuyen1 dia tu coc', c1, ' → ', c3)
Else
Begin
Chuyen(n-1,c1,c2,c3);
```


http://tieulun.hopto.org

```
    Chuyen(1,c1,c3,c2);
    Chuyen(n-1,c2,c3,c1);
    End;
    End;
```

Trong chương trình chính, sau khi nhập n đủ nhỏ, ta chỉ việc gọi thực hiện thủ tục: Chuyen(n,'A','C','B');

2. Bài toán chia phần

Có n gói quà và m người. Hãy viết chương trình tìm tất cả các cách chia n gói quà cho m người (m,n đủ nhỏ).

2.1. Quy nạp toán học

Ta dùng mảng a[1..n] of integer để biểu diễn một cách chia quà, trong đó $a[i] \leq 0$ là số gói quà của người thứ i.

Ta sẽ ưu tiên chia quà cho người cuối cùng. Nghĩa là ta cứ tiếp tục chia quà cho người cuối cùng khi còn gói quà. Khi không còn gói quà nào ta được một cách chia phần.

Để có các cách chia mới, ta sẽ bớt dần các gói quà của người cuối cùng, số gói quà được bớt ra đó sẽ chia cho người gần cuối cùng. Và, việc chia quà cho người người thứ n - 1 (người gần cuối cùng) được thực hiện giống như cách chia quà cho người thứ n (người cuối cùng), nghĩa là cho đến khi hết gói quà thì lại bớt dần các gói quà của người thứ n - 1 để ưu tiên chia cho người thứ n - 2. Và cách chia các gói quà được bớt ra cho người thứ n - 2 giống như cách chia quà cho người thứ n - 1. Cứ tiếp tục (một cách đệ quy như thế) cho đến khi xét đến việc ưu tiên chia quà cho người đầu tiên.

2.2. Phân tích thủ tục đệ quy

Gọi Chia_qua(m,n) là thủ tục thể hiện cách chia m gói quà cho n người.

Việc ưu tiên tiếp tục chia quà cho người cuối cùng và đến khi hết các gói quà thì in ra một cách chia được thể hiện như sau:

```
a[n] := a[n] +1;
If m>1 then Chia_qua(m-1,n)
Else InKq;
```

Sau khi in ra một cách chia, dừng đệ quy đối với người thứ n. Ta bớt dần các gói quà để ưu tiên chia các gói quà cho người ngay trước đó:

http://tieulun.hopto.org

a[n]:=a[n-1];

If n>1 then Chia_qua(m,n-1);

Trong hai phần đệ quy trên thì phần đệ quy thứ nhất được thực hiện nhiều hơn. Việc chia các gói quà cho người thứ n - 1 sẽ được thực hiện bởi phần đệ quy thứ nhất.

2.3. Thủ tục đệ quy

```
Procedure chia_qua(m,n: integer);
Begin
    a[n]: = a[n]+1;
    if m > 1 then chia_qua(m-1,n)
    else inkq;
    a[n]: = a[n]-1;
    if n > 1 then chia_qua(m,n-1) ;
End;
```

3. Bài toán phân tích số

Tìm tất cả các cách phân biệt phân tích số tự nhiên n thành tổng các số tự nhiên nhỏ hơn nó (n đủ nhỏ).

3.1. Quy nạp toán học

Ta biểu diễn một cách phân tích số tự nhiên n thành tổng các số nhỏ hơn nó trong mảng a: array[1..n] of integer;

Ví dụ: Với n = 4 thì hai cách phân tích sau đây là như nhau:

4 = 1+3

4 = 3+1

Vậy để tất cả các cách phân tích là phân biệt, ta có thể ràng buộc a[i] ≤ a[i+1]. Vậy ta có 4 cách phân tích phân biệt:

4 = 1+1+1+1

4 = 1+1+2

4 = 1+3

4 = 2+2

Việc phân tích số n có thể được hiểu như một quá trình đệ quy tách số n thành một số nào đó và phần bù của số đó (trong n).

http://tieulun.hopto.org

Đặt $n_1 = n$.

Đầu tiên số n_1 được tách thành số a[1] và phần bù của a[1]:

$n_1 = a[1] + (n_1 - a[1])$,

trong đó a[1] có thể nhận một trong các giá trị 1,2,....,n_1.

Với số $n_2 = n_1 - a[1]$ còn lại, ta có thể tách số n_2 giống như cách tách số n_1, nghĩa là n_2 được tách thành số a[2] và phần bù của a[2]:

$n_2 = a[2] + (n_2 - a[2])$,

trong đó a[2] có thể nhận một trong các giá trị 1,2,...,n_2.

Để ý rằng lúc này n đã tách được 2 số a[1] và a[2], nghĩa là:

$n = a[1] + a[2] + (n_2 - a[2])$

Tổng quát, đến "bước thứ k", ta có:

$n_k = a[k] + (n_k - a[k])$,

trong đó a[k] có thể nhận một trong các giá trị 1, 2, ...,n_k.

Tại bước này, số tự nhiên n đã tách được k số a[1], a[2],. .., a[k], nghĩa là:

$n = a[1] + a[2] +. .. + a[k] + (n_k - a[k])$,

Khi đó, số $n_{k+1} = n_k - a[k]$ còn lại có thể được tách giống như cách tách số n_k trước đó.

Cứ tiếp tục quá trình trên đến khi số còn lại = 0, nghĩa là không còn tách được nữa thì ta được một cách tách (phân tích) đầy đủ của n.

Một điều cần chính xác lại là trong phép tách số n_{k+1}:

$n_{k+1} = a[k+1] + (n_{k+1} - a[k+1])$, trong đó a[k+1] có thể nhận giá trị nhiều nhất là n_{k+1} và ít nhất là a[k], vì theo giả thiết có $a[i+1] \geq a[i]$.

Tổng quát, nếu trước đó số n còn lại đã tách được k số thì số a[k+1] cần tách sẽ có thể nhận các giá trị từ d đến n, trong đó d = 1 nếu k = 0 và d = a[k] nếu k > 0.

3.2. Phân tích thủ tục đệ quy

Gọi Tách(n,k) là thủ tục thực hiện việc tách số n thành số a[k+1] và phần bù của a[k+1], với điều kiện trước đó n đã tách được k số.

Việc tách sẽ là:

a[k+1]=d,d+1,...,n, với d = 1 nếu k = 0, và d = a[k] nếu k > 0

Ta tách phần còn lại n – a[k+1] giống như cách tách số n, nghĩa là thực hiện lời gọi thủ tục Tách(n - a[k +1],k+1).

http://tieulun.hopto.org

3.3. Thủ tục đệ quy

```
Procedure Tach(n,k: byte);
Var i,d: byte;
Begin
 If n=0 Then INKQ(k)
 Else
 Begin
  If k=0 Then d:=1
  Else d:=A[k];
  For i:=d to n do
  Begin
    A[k+1]:=i;
    Phantich(n-i,k+1);
  End;
 End;
End;
```

4. Bài toán tìm các hoán vị

Tìm tất cả các hoán vị của dãy n số tự nhiên đầu tiên.

4.1. Quy nạp toán học

Ta dùng mảng $a_1,a_2,...,a_n$ để biểu diễn một hoán vị của n số tự nhiên đầu tiên. Nói chung phương pháp quy nạp sau đây đúng trong trường hợp n phần tử bất kỳ.

Ta thấy rằng mỗi một hoán vị tiếp theo thu được bằng cách đổi chỗ 2 phần tử ở vị trí bất kỳ trong hoán vị trước đó. Tuy nhiên để tất cả các hoán vị sinh ra đều phân biệt ta sẽ quy định chặt chẽ hơn.

Với n = 1, đương nhiên ta có một hoán vị.

Với n = 2 ta chỉ cần đổi chỗ 2 phần tử để thu được hoán vị mới. Như vậy ta đã giải được bài toán với n = 2 và có 2 hoán vị.

Với n = 3 ta sẽ lần lượt đổi chỗ a_3 với a_2 và a_1 và sau khi đổi chỗ thì ta không cần quan tâm đến phần tử thức ba là a_3 nữa mà chỉ quan tâm 2 phần tử đầu tiên a_1 và a_2. Nhưng khi đó thì ta lại thu được bài toán đã giải với n = 2.

http://tieulun.hopto.org

Theo trên với n = 2 ta có 2 hoán vị và với 2 lần đổi chỗ ta sẽ có 2 x 2 = 4 hoán vị, vậy còn thiếu 2 hoán vị nữa. Cụ thể ta có:

Ban đầu là dãy:

1 2 3

Sau khi đổi chỗ a_3 và a_2 ta có 2 hoán vị:

1 3 2

3 1 2

Sau khi đổi chỗ a_3 và a_1 ta có 2 hoán vị:

2 3 1

3 2 1

Ta thấy thiếu 2 hoán vị sau đây:

1 2 3

2 1 3

có nghĩa là các hoán vị còn thiếu là các hoán vị mà giữ nguyên phần tử a_3 và thực hiện hoán vị 2 số a_1 và a_2 trước đó.

Để tránh thiếu các hoán vị loại này, ta cho thực hiện 3 phép đổi chỗ a_3 với a_3, a_2, $a_1$và với mỗi một phép đổi chỗ ta chuyển về bài toán tìm các hoán vị với n = 2 đã giải. Việc "dậm chân tại chỗ" doicho(3,3) sẽ giúp tìm các hoán vị thiếu trong phân tích trên. Vậy ta có 3 lần đổi chỗ x 2 = 6 hoán vị tất cả.

Tóm lại bài toán giải được với n = 3.

Một cách tổng quát, giả sử ta đã tìm được (n - 1)! hoán vị của n - 1 phần tử $a_1,a_2,..,a_{n-1}$. Ta có thể tìm được hoán vị của n phần tử $a_1,a_2,..,a_{n-1},a_n$ như sau: ta sẽ lần lượt thực hiện việc đổi chỗ a_n với các phần tử $a_n,a_{n-1},...,a_1$, với mỗi lần đổi chỗ như thế, không xét phần tử a_n ta sẽ đưa về bài toán tìm hoán vị của n - 1 phần tử đã giải. Vậy tổng số hoán vị sẽ là n lần đổi chỗ nhân với (n - 1)! hoán vị đã tìm và kết quả bằng n! hoán vị.

4.2. Phân tích thủ tục đệ quy

Gọi Hoánvị(n:integer) là thủ tục đệ quy tìm tất cả các hoán vị của n phần tử $a_1,a_2,...,a_n$. Theo phương pháp quy nạp trên ta có:

Quá trình đệ quy là quá trình giải các bài toán:

Hoánvị(n) ← Hoánvị(n-1) ←...

For k:=n downto 1 do

http://tieulun.hopto.org

```
Begin
 Doicho(a[n],a[k]);
 Hoanvi(n-1);
End;
```

Vậy phần neo của đệ quy chính là bài toán đơn giản nhất với n = 1. Nghĩa là tại đây ta thu được một hoán vị hoàn chỉnh:

```
if n=1 then Inkq;
```

4.3. Thủ tục đệ quy

```
Procedure Hoanvi(A:Arr; n: Integer);
  Var
   k,tg,i:Integer;
  Begin
   If n=1 then Inkq
   else
   For k:=n downto 1 do
   begin
     tg:=A[k];A[k]:=A[n];A[n]:=tg; If k>1 then Hoanvi(a,k-1);
   end;
  End;
```

Trong đó Arr =Array[1..100] of Byte; Và ban đầu mảng A khởi tạo A[i]=i, ∀ i=1,2,...,n.

4.4. Bàn về mặt kỹ thuật

Ta nên trả lời được 2 câu hỏi sau đây:

(1) Tại sao cần truyền cả mảng A vào thủ tục đệ quy, phải chăng chỉ cần khai báo và định nghĩa Hoanvi(n) là đủ?

(2) Tại sao phải for k:=n downto 1, có thể thay bằng for k:=1 to n thì có được không?

Trả lời câu 1:

Nếu không truyền A vào thủ tục đệ quy thì A trở thành biến toàn cục và việc lưu ngữ cảnh của các phép đổi chỗ 2 phần tử trong A không được lưu lại, do đó kết quả sẽ bị sai hoặc trùng lặp.

Chẳng hạn với n = 3 ta sẽ có kết quả bị sai như sau:

1 2 3 (1)

http://tieulun.hopto.org

2 1 3 (2)

2 3 1 (3)

3 2 1 (4)

1 2 3 (5)

2 1 3 (6)

Với dãy ban đầu là 1 2 3 thì việc đổi chỗ (3,3) cho ta 2 hoán vị (1) và (2). Khi kết thúc phần đệ quy địa phương này thì vì A là toàn cục nên lúc này A là kết quả của (2), nghĩa là:

2 1 3.

Bây giờ việc đổi chỗ (3,2) tức là đổi chỗ 3 và 1 cho nhau, sẽ cho 2 hoán vị (3) và (4). Tương tự như trên, khi kết thúc phần đệ quy địa phương này thì vì A là toàn cục nên lúc này A là kết quả của (4), nghĩa là:

3 2 1

Bây giờ việc đổi chỗ (3,1) tức là đổi chỗ 3 và 1 cho nhau, sẽ cho 2 hoán vị (5) và (6).

Vậy kết quả trên sai vì thiếu và thừa (có sự trùng lặp), và nói chung là quá trình đệ quy diễn ra vô nghĩa!

* Nếu A được truyền vào thủ tục đệ quy thì ta có 6 hoán vị đúng sau đây:

1 2 3 (1)

2 1 3 (2)

1 3 2 (3)

3 1 2 (4)

2 3 1 (5)

3 2 1 (6)

Với dãy ban đầu là 1 2 3 thì việc đổi chỗ (3,3) cho ta 2 hoán vị (1) và (2). Khi kết thúc phần đệ quy địa phương này thì ngữ cảnh cũ được khôi phục lại, tức là A bằng:

1 2 3

Bây giờ việc đổi chỗ (3,2) tức là đổi chỗ 3 và 2 cho nhau, sẽ cho 2 hoán vị (3) và (4). Tương tự như trên, khi kết thúc phần đệ quy địa phương này thì ngữ cảnh cũ được khôi phục lại, tức là A bằng:

1 2 3

http://tieulun.hopto.org

Bây giờ việc đổi chỗ (3,1) tức là đổi chỗ 3 và 1 cho nhau, sẽ cho 2 hoán vị (5) và (6).

Nếu ta không muốn truyền A vào thủ tục đệ quy thì ta cần viết thủ tục đệ quy dưới dạng thủ tục Backtracking (quaylui):

```
Procedure Hoanvi(n: Integer);
  Var
   k,tg,i:Integer;
  Begin
   If n=1 then Inkq
   else
   For k:=n downto 1 do
   begin
      tg:=A[k];A[k]:=A[n];A[n]:=tg;
      If k>1 then Hoanvi(a,k-1);
      tg:=A[k];A[k]:=A[n];A[n]:=tg;
   end;
End;
```

Và khi đó, ta có thể for k:=1 to n.

Trả lời câu 2:

Nói một cách nôm na thì thủ tục đệ quy của ta không những đệ quy theo số lượng mà còn đệ quy theo thứ tự. Nghĩa là bài toán Hoánvị(n) chuyển thành bài toán Hoánvị(n - 1) với giả thiết bài toán tìm tất cả các hoán vị của n - 1 phần tử trong dãy $a_1,a_2,..,a_i$ đã giải được. Có nghĩa là bài toán được quy về trong lời gọi đệ quy là đi từ phải sang trái.

5. Câu hỏi và bài tập

Bài PL2.1: Tìm số lượng cách chia quà

Tìm số lượng cách chia m gói quà cho n người sao cho người trước nhận được số quà không ít hơn số quà của người sau.

Các số nguyên dương m, n ≤ 20.

Bài PL2.2: Tìm các phân tích Fibonaci

Mọi số tự nhiên n đều có thể biểu diễn dưới dạng tổng của các số Fibonaci, trong đó không có 2 số fibonaci đứng liền nhau:

http://tieulun.hopto.org

n = a[1]*x[1] + a[2]*x[2] +. .. + a[k]*x[k]

Trong đó:

- x[i] là số fibonaci thứ i.
- a[i] = 0 hoặc 1 với ∀ i =1,2,....,k
- Không tồn tại i =1,2....,k-1 sao cho a[i] = a[i+1] = 1

Hãy tìm một cách phân tích số tự nhiên n<=30000 dưới dạng trên.

Bài PL2.3: Chia đoạn thẳng

Có một đoạn thẳng nằm trên trục hoành, một đầu tại gốc toạ độ, đầu kia có toạ độ K (nguyên). Người ta chia đoạn thẳng thành 3 phần bằng nhau và gạch bỏ đoạn ở giữa nhưng vẫn để lại 2 đầu mút. Hai đoạn thẳng không bị gạch bỏ đánh số thứ tự là 1 và 2. Đối với 2 đoạn còn lại này lại được chia thành 3 phần bằng nhau giống như cách làm của đoạn đầu tiên. Như vậy sau khi chia xong thì số hiệu của 4 đoạn thẳng còn lại là 1, 2, 3 và 4. Có m lần chia như vậy.

Viết chương trình nhập k, m. Nhập tiếp từ bàn phím một phân số đúng x/y (x,y nguyên) và cho biết x/y có thuộc vào một trong các đoạn thẳng không bị gạch bỏ không? Nếu có thì đó là đoạn thẳng có số hiệu nào?

Bài PL2.4: Phủ thanh thước thợ

Một lưới G kích thước m x m (m = 2n, n < 50) gồm m^2 ô vuông cạnh 1 đơn vị. Người ta muốn xếp các thanh thước thợ để phủ kín các ô của lưới:

Chứng minh rằng luôn tồn tại một cách phủ m^2 - 1 các ô của lưới.

Nhập vào từ bàn phím số nguyên dương n. Hãy chỉ ra một cách phủ các ô của lưới sao cho chỉ để lại một ô không được phủ.

http://tieulun.hopto.org

HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP

I. HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP CHƯƠNG 1

Bài 1.2

Bộ dữ liệu kiểm thử đầy đủ cho bất phương trình ax + b >=0 gồm:

a = 0, b = 0

a = 0, b >= 0

a = 0, b = -4 <0

a =5 > 0, b = 8 (hoặc giá trị bất kỳ của a > 0, và b)

a = -5 <0, b = 9 (hoặc giá trị bất kỳ của a < 0 và b)

Bài 1.3

Đó là thuật toán tổng quát đổi giá trị 2 biến a và b có kiểu dữ liệu đơn giản.

Bảng mô phỏng như sau:

Bước	a	b	x
1	7	9	chưa xác định
2	7	9	7
3	9	9	7
4	9	7	7
5	9	7	

Bài 1.5

Để tìm ước số chung lớn nhất của (a,b) = (100,10) thì thuật toán trừ liên tiếp thực hiện 100-10 = 90 lần. Thuật toán chia liên tiếp thực hiện 0 lần, vì ngay từ đầu số dư của a chia cho b bằng 0, vòng lặp không thực hiện, b chính là ước chung lớn nhất cần tìm.

Bài 1.6

Tính P=n!

http://tieulun.hopto.org

Phương pháp liệt kê từng bước

Bước 1. Nhập n;

Bước 2. Nếu n < 0 thì làm Bước 1;

Bước 3. P:=1; i:=0;

Bước 4. Nếu i>=n thì làm Bước 6;

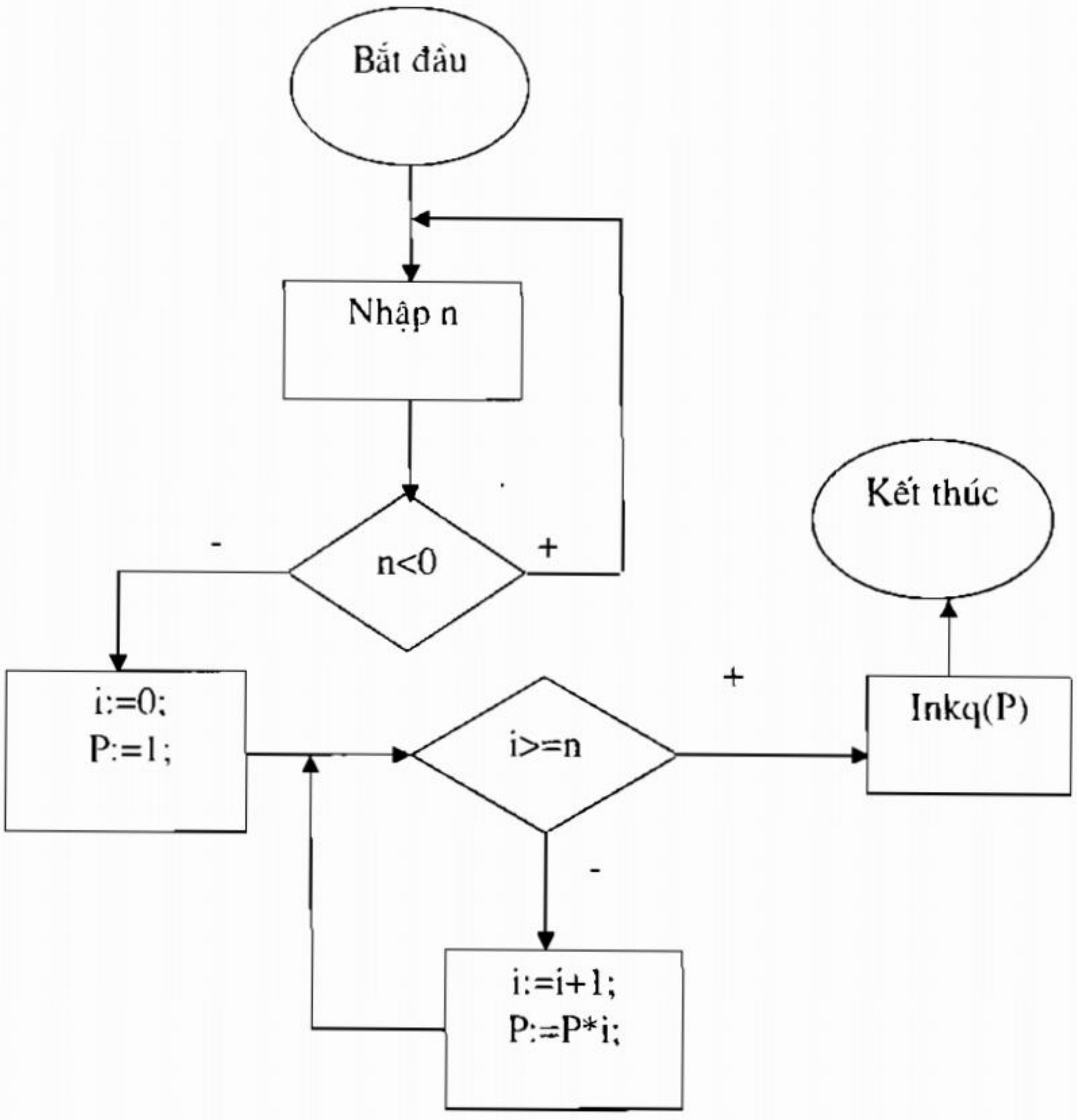
*Bước 5. i:= i + 1; P:= P * i;*

Bước 6. Quay về Bước 4;

Bước 7. In kết quả (P);

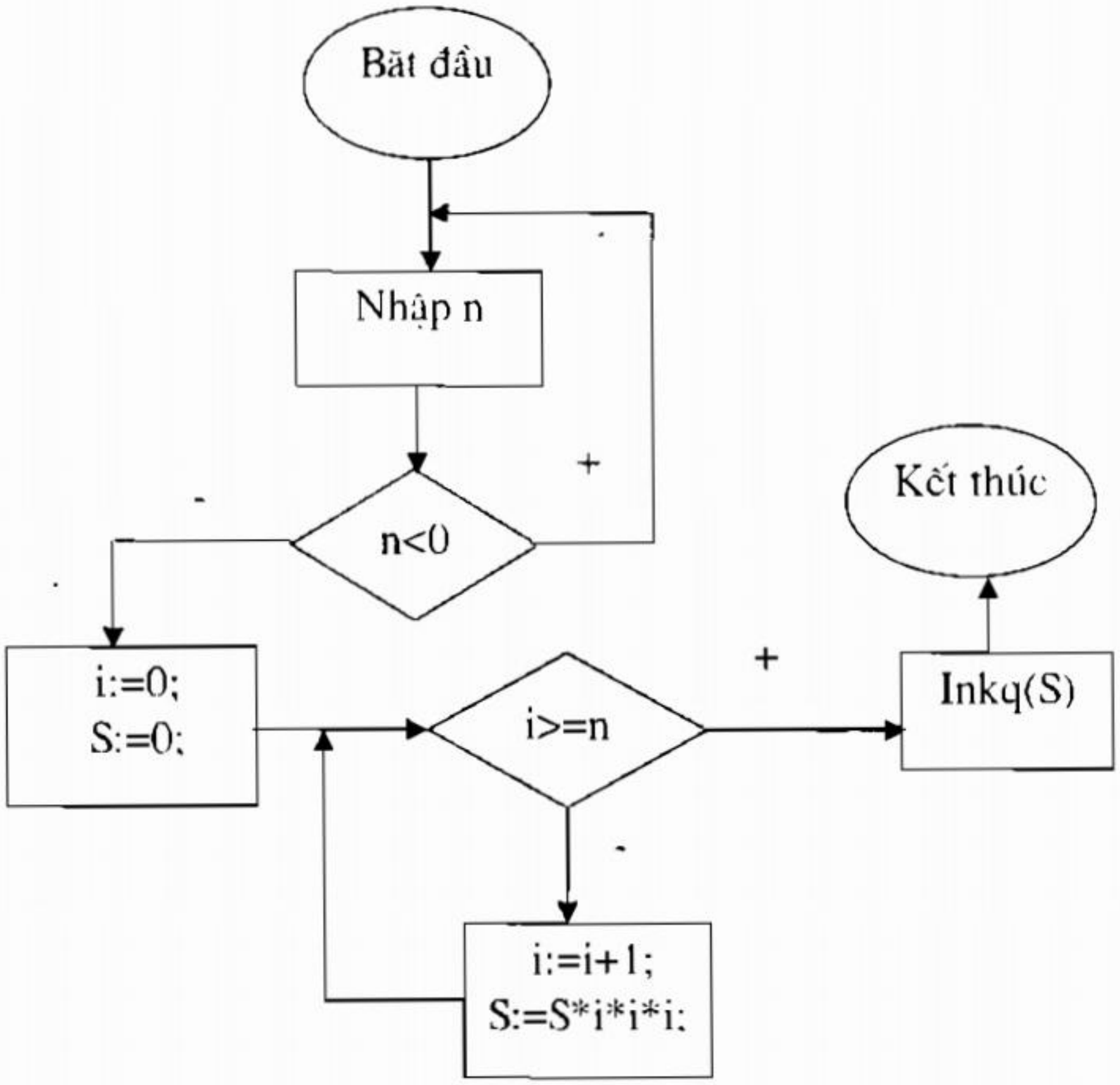
Bước 8. Kết thúc.

Phương pháp dùng sơ đồ khối



http://tieulun.hopto.org

Bài 1.7



Bài 1.8

Bước 1. i:= 1;
Bước 2. j:= 0;
Bước 3. In ra (i,j)
Bước 4. j:= j+1;
Bước 5. Nếu j <=9 thì làm Bước 3;
Bước 6. i:= i+1;
Bước 7. Nếu i <= 9 thì làm Bước 2;
Bước 8. Kết thúc.

II. HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP CHƯƠNG 2

Bài 2.1

1) Chạy từ dấu nhắc MS-DOS.

http://tieulun.hopto.org

2) Chạy từ NC.

3) Chạy từ Window Explore.

Tất cả các cách trên: Chuyển vào thư mục chứa các files TURBO.*, sau đó gọi thực hiện chương trình TURBO.EXE.

4) Trong windows 9.x có thể chạy TURBO.EXE từ lệnh Run và chú ý Properties cho file TURBO.EXE (nhấn chuột phải vào file TURBO.EXE chọn Properties) như sau:

- Tự động đóng: Close on exit.
- Tương thích môi trường window.

Bài 2.2

- Lỗi cú pháp là lỗi khi dịch chương trình do viết sai quy tắc của ngôn ngữ, của cấu trúc lệnh.Ví dụ lỗi cú pháp xảy ra khi sau lệnh thiếu dấu chấm phẩy (;), trước else có dấu chấm phẩy, có begin mà không có end tương ứng, có repeat mà không có until tương ứng....
- Lỗi thuật toán là lỗi khi chạy chương trình và cho kết quả sai, thuật toán vi phạm tính xác định. Trường hợp này phải sửa lại để thuật toán đúng đắn.

Bài 2.4

```
Const Pi = 3.1416;
Var r,cv,dt: real;
Begin
 Write('Nhap ban kinh: '); Readlnđ;
 cv:= 2*pi*r;
 dt:= pi*r*r;
 Writeln('Chu vi hinh tron la: ', cv:10:3);
 Writeln('Dien tich hinh tron la: ', dt:10:3);
 Readln;
End.
```

Bài 2.5

```
Var a,b,c: real;
        p, S: Real;
Begin
Write('Nhap 3 canh cua tam giac: '); Readln(a,b,c);
```


http://tieulun.hopto.org

```
p:= (a+b+c)/2;
S:= sqrt(p*(p-a)*(p-b)*(p-c));
Writeln('Chu vi ', p: 10:2);
Writeln('Dien tich: ', S: 10: 2);
Writeln('ha = ', 2*S/a: 10: 2);
Writeln('hb = ', 2*S/b:10: 2);
Writeln('hc = ', 2*S/c: 10:2 );
Writeln('ma = ', (1/2)* sqrt (2*b*b + 2*c*c -a*a):10:2);
Writeln('mb = ', (1/2)* sqrt (2*a*a + 2*c*c -b*b):10:2);
Writeln('mc = ', (1/2)* sqrt (2*a*a + 2*b*b -c*c):10:2);
Writeln('la = ', (2/(b+c))*sqrt(b*c*p*(p-a)):10:2);
Writeln('lb = ', (2/(a+c))*sqrt(a*c*p*(p-b)):10:2);
Writeln('lc = ', (2/(b+c))*sqrt(a*b*p*(p-c)):10:2);
Writeln('R ngt = ', a*b*c/4*s:10:2);
Writeln('R nt = ', S/p:10:2);
Readln;
End.
```

Bài 2.7

- Lệnh a:= a/b; sai vì vế trái là một biến integer không thể nhận giá trị kiểu real của vế phải.

- Lệnh writeln(r mod 2); sai vì phép mod không áp dụng được cho biến kiểu real.

- Lệnh writeln(b:10:3); sai vì biến nguyên b không có phần thập phân trong quy cách in.

Bài 2.8

- Lệnh Write(Char(65),'nh', #13, #10, 'Binh'); in lên màn hình 2 dòng:

Anh
Binh

- Lệnh Write(34>56); in lên màn hình chữ FALSE.

Bài 2.9

Lệnh	Giá trị của biến boolean
x:=9; m:=101; error:= (x <0) or (x>10);	 error = false or false = false

http://tieulun.hopto.org

good:= Not Error;	good = not false = true
ok:= (m >= 10) and (m <= 99);	ok = true and false = false
stop:= good or ok;	stop = true or false = true

Bài 2.10

```
Var a,b,c: real;
Begin
Write('Nhap cac he so: '); Readln(a,b,c);
If a = 0 then
 If b = 0 then
   If c = 0 then Writeln('PT vo so nghiem')
   Else Writeln('PT vo nghiem')
 Else {  a=0, b <> 0}
 Writeln('PT co nghiem x = ', -c/b: 10: 3)
Else { a <> 0 }
Begin
 c:= b*b-4*a*c;
 a:= 2 *a;
 If c < 0 then Writeln('PT vo nghiem')
 Else
 If c = 0 then Writeln('PT co nghiem kep x0 = ', -b/a: 10:3);
 Else
 Begin
   Writeln('PT co 2 nghiem phan biet: ');
   Writeln('x1 = ', (-b + sqrt(c))/a: 10: 3);
    Writeln('x2 = ', (-b - sqrt(c))/a: 10: 3);
  End;
End;
Readln;
End.
```


http://tieulun.hopto.org

Bài 2.12

```
Var cst, csn, cstt: integer;
    tt: longint;
Begin
 Write('Nhap chi so dien thang truoc: '); Readln(cst);
 Write('Nhap chi so dien thang nay: '); Readln(csn);
 cstt:= csn-cst;
 If cstt <=100 Then tt:= cstt * 400
 Else If cstt <= 150 Then
        tt:= 100 * 400 + (cstt-100) * 500
  Else If cstt <= 200 Then
        tt:= 100 * 400 + 50 * 500 + (cstt - 150) * 800
  Else
  tt:=100 * 400 + 50 * 500 + 50 * 800 + (cstt-200) * 1000;
Writeln('Tien dien phai thanh toan la: ', tt: 10: 3);
Readln;
End.
```

Bài 2.14

```
Var a,b,x,y,s,i: longint;
Begin
Write('a,b = '); Readln(a,b);
 For x:= a to b do
 Begin
  y:=0; { y la tong cac uoc cua x}
  For i:=1 to x div 2 do
  If x mod i = 0 then y:=y+i;
  s = 0; { s la tong cac uoc cua y}
  For i:=1 to y div 2 do
  If y mod i = 0 then s:= s + i;
    If s=x then Writeln(x,',' ,y);
End;
Readln;
End.
```


http://tieulun.hopto.org

Bài 2.16

```
Var g,c: integer;
Begin
 For g:= 0 to 36 do
 For c:= 0 to 36-g do
 If (g+c=36) and (g*2 + c *4) = 100 then
Writeln('So con Ga: ',g,' So con cho: ',c);
Readln;
End.
```

Bài 2.20

- Gọi b là số lượng người trong các năm.
- Ban đầu b:=a; số năm cần tính n:= 0;
- Lặp quá trình sau đây khi b < p* a.

+ Cứ sau mỗi năm dân số là:

b:= b + b * k/100; (lấy b là biến real)

+ Và thêm được 1 năm:

n:= n + 1;

Bài 2.21

- Khởi tạo M:=0; i:=1;
- Lặp 2 lệnh sau đây khi M <= 2*10E+6

M:= M + i*i;

i:= i + 1;

Bài 2.22

Gợi ý:

Gọi r = n mod (k + 1), n là số sỏi còn lại sau mỗi lượt bốc. Máy bốc với số lượng sỏi bằng a như sau: nếu r = 0 thì a=k, nếu r<>0 và r <> 1 thì a = r-1, nếu r=1 thì máy bốc ngẫu nhiên (thường là 1 viên).

III. HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP CHƯƠNG 3

Bài 3.1

Khai báo mảng điểm:

const maxN=60;

http://tieulun.hopto.org

```
Var A: array[1..maxn] of real;
  n: integer;
```

Câu 1: Áp dụng mẫu nhập dữ liệu cho mảng 1 chiều trong bài học.

Câu 2:

```
Bước 1: Tính max
max:= a[1];
for i:= 2 to n do
If a[i]>max then max:= a[i];
In ra màn hình giá trị max
Bước 2: In ra i mà a[i] = max
For i:= 1 to n do
If a[i]=max then write(i, #32);
```

Câu 3: Áp dụng một thuật toán sắp xếp đã học trong bài đối với mảng a, chẳng hạn chọn sắp xếp kiểu chọn trực tiếp. Tuy nhiên, trong bài tập này thứ tự sinh viên (tức là thứ tự các phần tử trong mảng a) được xem như thay cho tên học sinh. Sau khi sắp xếp điểm sinh viên lưu trong mảng a thì tên học sinh và điểm của họ không nhất quán như ban đầu. Để đảm bảo sự nhất quán giữa tên và điểm sau khi xáo trộn mảng, ta làm như sau:

Dùng thêm biến mảng T để biểu thị tên học sinh (số thứ tự):

T: array[1..maxn] of integer;

Vậy 2 mảng a và T là 2 mảng song song với nhau tại chỉ số. Học sinh thứ i có tên là T[i], có điểm là a[i].

Trước khi sắp xếp ta khởi gán T[i]:= i với i=1,2,...,n

For i:=1 to n do T[i]:=i;

Trong khi sắp xếp mảng a, nếu có sự đổi chỗ 2 phần tử a[i] và a[j] thì đồng thời ta cũng đổi chỗ 2 phần tử T[i] và T[j].

```
For i:=1 to n-1 do
For j:=i+1 to n do
If a[j] > a[i] then
Begin tg1:= a[i]; a[i]:= a[j]; a[j]:= tg1;
      tg2:= T[i]; T[i]:= T[j]; T[j]:= tg2;
End;
```


http://tieulun.hopto.org

Lưu ý kiểu dữ liệu của tg1 là real, của tg2 là integer.

Câu 4: Ta chỉ cần duyệt lại mảng a từ cuối về đầu. Với mỗi i = n,n-1,..., 1, ta in ra T[i] và a[i].

Bài 3.2

Câu 2:

For i:=1 to n do

Begin { xet a[i]}

- Áp dụng thuật toán kiểm tra tính nguyên tố cho số a[i].
- Nếu a[i] là nguyên tố thì in ra a[i];

End;

Bài 3.3

Gọi kg là số lượng dãy không giảm (<=) và kt là số lượng dãy không tăng (>=)

Dễ dàng tính được kt và kg:

```
kg:= 0; kt:= 0;
For t:=1 to n-1 do
Begin
 If a[i] <= a[i+1] then inc(kg);
 If a[i] >= a[i+1] then inc(kt);
End;
```

Như vậy có khả năng sau:

- Nếu kg=kt=n-1 thì a là dãy dừng. Nếu không thì:
- Nếu kg=n-1 thì a là dãy không giảm. Nếu không thì:
- Nếu kt=n-1 thì a là dãy không tăng. Nếu không thì a không thoả mãn 1 trong 3 tính chất đầu tiên.

Tính chất cấp số cộng nên kiểm tra riêng.

```
Đặt d = a[1] - a[2];
For i:=2 to n-1 do
If a[i] - a[i+1] <> d then
Begin
   d:=0; Break;
End;
```


http://tieulun.hopto.org

Vậy nếu d<>0 thì a là một cấp số cộng với công sai là d.

Bài 3.4

Tổ chức dữ liệu chính:

```
Type TenHang = 'A'..'Z';
Var a: array[TenHang] Of integer;
     n: integer;
     i, th1, th2: TenHang;
```

Với n <=26, mặt hàng đầu tên là th1:=chr(65), mặt hàng cuối cùng tên là th2:= chr(64+n);

Câu 1:

```
For i:= th1 to th2 do
Repeat
  Write('a[',i,']= '); Readln(a[i]);
Until a[i]>0;
```

Chú ý việc duyệt trên mảng a từ đây có dạng for i:=th1 to th2 do....

Câu 2:

Trước hết tính điểm trung bình tb bằng cách tính tb là tổng của các phần tử trong mảng a, sau đó gán tb:= tb/n. Lưu ý tb có kiểu real. Cuối cùng ta duyệt lại mảng a, tại i mà a[i] >=tb thì in i ra màn hình.

Câu 3:

Thứ tự các mặt hàng được xem như tên hàng. Hãy xem lại kỹ thuật đồng bộ tên hàng và số lượng hàng khi xáo trộn các mặt hàng trong mảng lúc sắp xếp mảng ở câu 3 bài tập 3. 1

Bài 3.5

Có thể in ra màn hình dạng:

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Hàng thứ i có i + 1 số, có tất cả n hàng.

Ta dùng 2 mảng một chiều kích thước n: a và b. Không phải bất kỳ lúc nào ta cũng dùng hết n phần tử. Mảng a để biểu diễn một hàng thứ i nào đó, mảng b để biểu diễn hàng thứ i+1, bên dưới hàng i.

http://tieulun.hopto.org

Nhận xét: Nếu mảng a đã biết (ở hàng i-1) thì mảng b có thể tính được (tại hàng i) như sau:

b[1]=b[i]=1:

b[j] = a[i-1] + a[i]: với 2<=j<=i-1

Vậy việc in ra tam giác Pascal như sau:

```
+ Khởi gán a[1]:=1; a[2]:=1; In ra số 1.
+ For i:=2 to n do
Begin
{Tinh hàng i dựa vào hàng i-1}
     b[1]:=1; b[i]:=1;
     For j:=2 to i-1 do b[j]:=a[i-1]+a[i];
{In ra hàng i}
For j:=1 to i do write(b[j]:4); writeln;
End;
```

Bài 3.6

Dùng mảng một chiều a kích thước 30 biểu diễn cho số đó có n chữ số (n<=30).

a) Kiểm tra đối xứng:

```
dx:= true;
For i:=1 to n div 2 do
If a[i] <> a[n-i+1] then dx:= false;
```

Đến đây, số đó là đối xứng hay không phụ thuộc vào giá trị true hay false của biến dx.

b) Để kiểm tra số đó có chia hết cho 3 ta tính tổng giá trị các phần tử của mảng a rồi dùng hàm mod.

Bài 3.8

Thực hiện việc chèn x như sau:

+ Nếu x <= a[1] ta tịnh tiến mảng a sang phải một vị trí và đặt a[1]:=x;

+ Nếu x >= a[n] ta a[n+1]:=x;

+ Nếu không rơi vào hai trường hợp trên, tìm vị trí i trong mảng a mà a[i]<=x<=a[i+1]. Tịnh tiến các phần tử của mảng a từ vị trí i+1 sang phải một vị trí rồi thực hiện chèn x: a[i+1]:=x;

http://tieulun.hopto.org

Đoạn trình tịnh tiến mảng a sang phải bắt đầu từ vị trí k như sau:

```
for j:=n downto k do a[j+1]:= a[j];
```

Bài 3.9

Tổ chức dữ liệu chính:

```
const maxm = 20; maxn = 30;
Var a: array[1..maxm, 1..maxn] of integer;
    m,n,i,j: integer;
```

m hàng của mảng biểu thị m điểm phát, n cột của biểu thị các điểm thu.

Câu 2: Áp dụng mẫu duyệt mảng 2 chiều cố định hàng 1 và hàng m để tính tổng s1 và s2 tương ứng với 2 hàng đó. In ra màn hình abs(s1-s2);

Câu 3: Áp dụng mẫu duyệt toàn bộ mảng 2 chiều: "ưu tiên duyệt trên cột trước" và kỹ thuật 3 bước.

Trước hết phải tính max:

```
max:= 0;
For j:=1 to n do
Begin s:=0; For i:=1 to m do s:= s + a[i,j];
      if s > max then max:= s;
End;
```

In ra max và phải duyệt lại để in ra các điểm thu đạt tổng s bằng max.

```
For j:=1 to n do
Begin s:=0; For i:=1 to m do s:= s + a[i,j];
      if s = max then write(j, #32);
End;
```

Bài 3.10

Tổ chức dữ liệu chính:

```
const maxm = 25; {maxn = 20;}
Var a: array[1..maxm, 'a'..'z'] of real;
    m,n,i: integer; j: char;
```

Trong mảng a: m hàng biểu thị cho m sinh viên, không quá 26 cột biểu thị cho n đề tài.

http://tieulun.hopto.org

Câu 2: Với mỗi cột j, đếm số lượng phần tử bằng -1 để biết cột nào có số lượng số -1 ít nhất. Vậy, áp dụng mẫu duyệt toàn bộ mảng 2 chiều: "ưu tiên duyệt trên cột" và kỹ thuật 3 bước. Xem gợi ý câu 3 bài tập 3.1.

Câu 3, Câu 4: áp dụng mẫu duyệt toàn bộ mảng 2 chiều: "ưu tiên duyệt trên hàng" và kỹ thuật 3 bước. Xem gợi ý câu 3 bài tập 3.1.

Bài 3.11

Tổ chức dữ liệu:

```
const maxm=10; maxn=20;
Var  c: array[1..maxm, 1..maxn] of integer;
     nhth: array[1..maxm] of string;
     hm: array[1..maxn] of string;
     m,n: intger;
```

Câu 1, Câu 2: Nhập m,n và 3 mảng c, nhth, hm.

Câu 3, Câu 4: Với mỗi hàng (nhà thầu) , duyệt trên hàng đó (các hạng mục công trình), tìm vị trí một phần tử có c[i,j] nhỏ nhất. Giả sử vị trí đó là (i1,j1) in ra nhth[i1], hm[j1]. Tích luỹ C[i1,j1] vào tổng S cho câu 4.

Bài 3.14

Giá trị của k không thay đổi khi s không thay đổi và bằng vị trí xuất hiện lần đầu tiên của xâu Anh trong S, k =7.

Bài 3.16

Phép cộng xâu không có tính chất giao hoán.

Phép so sánh 2 xâu ký tự có đảm bảo về thứ tự từ điển vì đó là phép so sánh 2 mã ASCII của 2 ký tự khác nhau đầu tiên trong 2 xâu.

```
Var A: array[1..255] of char;
```

Có thể nhập/in ra toàn bộ xâu S, trong khi mảng A phải nhập/in ra từng phần tử.

Mảng A không áp dụng được các hàm và thủ tục trên xâu.

Các phép duyệt trên S và A là giống nhau.

Bài 3.17

Khai báo 2 mảng một chiều HD (họ đệm) và T (tên) các phần tử kiểu string.

Câu 1: Sắp xếp mảng T như sắp xếp một mảng kiểu nguyên, khi đổi chỗ phần tử T[i] và T[j] thì cũng đổi chỗ 2 phần tử HD[i] và HD[j].

http://tieulun.hopto.org

Câu 2:

```
For i:=2 to n do
For j:=n downto i do
If (T[j] <T[j-1]) OR ( (T[j]=T[j-1]) and
(HD[j] < HD[j-1]) Then
Begin
   tg:= T[j]; T[j]:=T[j]-1; T[j-1]:= tg;
   tg:= HD[j]; HD[j]:=HD[j]-1; HD[j-1]:= tg;
End;
```

Bài 3.18

Giả thiết mỗi một từ không bị ngắt xuống dòng, các từ ngăn cách nhau bằng đúng một dấu cách, cuối mỗi câu có một dấu chấm câu.

Sử dụng một mảng các nguyên âm:

```
Type NguyenAm = ('A','E','O','U','I');
Var a: array [NguyenAm] of Integer;
     k: NguyenAm;
```

Khởi tạo:

```
sotu:=0; socau:=0; a[k]:=0; với mọi k = 'A' ..'I'
```

Quy ước: a[k] là số lượng nguyên âm k xuất hiện trong văn bản.

Dùng vòng lặp for i:=1 to n để nhập dòng S tất cả n lần. Xét dòng thứ i, dùng biến j để duyệt trên S:

+ Đếm được: số dấu cách d, số dấu chấm c.

+ Tích luỹ: sotu:= sotu + d + 1; socau:= socau + c;

+ Với mỗi ký tự S[j] trong phép duyệt trên xâu S: dùng lặp for k ='A' ...'I' để duyệt trên mảng a, nếu tồn tại k mà S[j] = k thì ta đếm thêm một lần xuất hiện nguyên âm k bằng cách tăng a[k] lên 1 đơn vị.

Sau n lần nhập:

+ In ra sotu, socau.

+ In ra nguyên âm k mà a[k] có giá trị lớn nhất.

Bài 3.19

- Giả xử X,Y là 2 xâu số có lỗi, a,b là 2 số sau khi sửa lỗi, c là tổng của a và b. Theo đầu bài a,b,c có kiểu longint.

http://tieulun.hopto.org

- Việc sửa lỗi X và Y như sau để tính a và b như sau:

```
Repeat Val(X,a,code); Until code=0;
Repeat Val(Y,b,code); Until code=0;
```

- Vậy c:= a + b;

Bài 3.23

- Không thể nhập giá trị cho biến kiểu liệt kê và kiểu khoảng con bình thường như các biến có kiểu dữ liệu đơn giản chuẩn.

- Kiểu đoạn con trong Pascal có thể biểu diễn một trục số có 256 điểm nguyên.

- Không thể khai báo một biến tập hợp kiểu dữ liệu có quá 256 phần tử, do đó không thể có kiểu nguyên.

Bài 3.24

A:=0; sai, vì A không là biến kiểu số (integer, real, byte,...).

A:=''; sai, vì A không là biến kiểu char hay string.

A:=[]; đúng, đây là lệnh khởi tạo tập rỗng.

A:=5; sai về cú pháp

A:=[5]; đúng, đây là lệnh khởi gán tập A.

A:= A + [5..9]; đúng, đây là phép hợp 2 tập hợp trong ngôn ngữ Pascal.

Bài 3.25

- Khai báo kiểu và biến:

Type TapDetai = Set Of Bye;

Var A,B,A1,B1: TapDetai;

- Nhập n,m thoả mãn n<=200.

- Gọi A là tập các đề tài mà tất cả các sinh viên đều viết, B là tập các đề tài mà không có sinh viên nào tham gia.

- A1,B1 là các biến tập hợp trung gian. A1 dùng để ghi nhận tạm thời tập các đề tài của từng sinh viên trong quá trình nhập. B1 là tập hợp ghi lại hợp của tất cả các tập đề tài mà các sinhviên tham gia.

- Khởi tạo A:=[1..n]; B1:=[];

- Nhập m lần cho các phần tử tập A1, sau mỗi lần nhập A1 được khởi tạo lại là tập rỗng. Lần nhập thứ i tương ứng với các đề tài của sinh viên thứ i tham gia. Sau khi nhập xong A1 ta có:

http://tieulun.hopto.org

A:= A * A1; A1:=[];

B1:= B1 + A1;

- Kết thúc quá trình nhập thì A là tập các đề tài mà tất cả các sinh viên đều tham gia. B = [1..n]-B1 là tập các đề tài còn lại, chưa có ai tham gia.

Bài 3.26

Câu 1: Giả sử n = 14, m=6

LOP={A,B,C,D,E,F,G,H,I,J,K,M,L,N}

6 quan hệ giả sử được nhập là:

AB → CD

CFK →MN

BGC →EF

CH →IK

J →LG

ADE → KH

Câu 2: Giả sử X=[A,B,G]

Khi đó X rủ được nhóm:

Y=[A,B,G,C,D,E,F,K,H,I,M,N]

Câu 3: Nhóm nòng cốt là Z=[A,B, J]

Gợi ý:

Về tổ chức dữ liệu chính.

Ta định nghĩa 2 kiểu dữ liệu:

```
TenHS = 'A' ..'Z'
NhomHS = Set Of TenHS;
```

Khi đó, ta dùng 2 mảng một chiều L và R trong đó L[i] là tập hợp biểu diễn vế trái của quan hệ thứ i, R[i] là tập hợp biểu diễn vế phải của quan hệ thứ i này, i = 1,2,..., m.

```
L,R: Array[1..MaxN] Of NhomHS;
(const MaxN = 26)
```

Nếu tên học sinh đầu và học sinh cuối được lưu trong 2 biến hsd và hsc thì ta có:

```
hsd:='A'; hsc:=Chr(64+n);
```


http://tieulun.hopto.org

Các phép duyệt trên các tập hợp S kiểu NhomHS sẽ có dạng:

```
For hs:=hsd to hsc do <Xử lý i đối với tập S>
```

Với hs là biến kiểu TenHS.

Thuật toán câu 2

Gọi Y là nhóm học sinh mà X rủ được.

Bước 1: Ban đầu Y:= X;

Bước 2: Xét mọi quan hệ i (i=1,2,...,m) nếu vế trái của quan hệ thứ i nào đó thuộc vào Y thì theo định nghĩa quan hệ L[i]→R[i], rõ ràng vế phải của quan hệ thứ i đó cũng phải thuộc vào Y. Để ngắn gọn trong lập luận, trong trường hợp này ta nói rằng "quan hệ i được thoả".

Hơn nữa, nếu quan hệ i được thoả thì ta ghi nhận có xảy xa việc kết nạp R[i] vào Y, đồng thời đánh dấu quan hệ này đã được xét.

```
Happened:= False;
For i:=1 to m do
Begin
 {kiểm tra xem L[i] có chứa trong Y không}
 Belong:= True;
 for k:= hsd to hsc do
 if not (k in Y) then
 Begin Belong:= false; Break;
 End;
 {Nếu L[i] chứa trong Y thì R[i] cũng chứa trong Y}
 If Belong and not ( i in DX) then
 Begin
 Y:= Y + R[i];
 DX:= DX + [i];
 Happened:= True;
End;
```

Bước 3: Nếu Happened = True thì ta quay về Bước 2. Ngược lại thì kết thúc thuật toán câu 2 và in ra các học sinh trong tập Y.

Tại sao phải quay lại bước 2?

Sở dĩ phải quay lại xét m quan hệ vì có thể khi quan hệ thứ j bên dưới được thoả dẫn đến quan hệ thứ j' nào đó bên trên được thoả, trong khi, trước đó, quan hệ thứ j' này không được thoả.

http://tieulun.hopto.org

Phân tích ví dụ đầu bài:

(1) AB → CD

(2) CFK →MN

(3) BGC →EF

(4) CH →IK

(5) J →LG

(6) ADE → KH

X=[A,B,G]

Bước 1: Y =[A,B,G]

Bước 2:

Quan hệ (1) được thoả, Y = [A,B,G,C,D].

Quan hệ (2) không được thoả.

Quan hệ (3) được thoả, Y = [A,B,G.C,D,E,F].

Quan hệ (4) không được thoả.

Quan hệ (5) không được thoả.

Quan hệ (6) được thoả, Y = [A,B,G,C,D,E,F,K,H].

Quay về xét lại Bước 2, thấy lúc này các quan hệ 2,4 được thoả.

Y=[A,B,G,C,D,E,F,K,H,I,M,N]

Quay về xét lại lần nữa, quan hệ 5 vẫn không được thoả, happened=false, dừng thuật toán.

Thuật toán câu 3

Thực ra câu 3 rất dễ thực hiện dựa vào câu 2 đã làm.

Ta có Calop=[hsd..hsc];

Gọi K là nhóm học sinh nòng cốt.

Ban đầu khởi tạo K:= Calop;

Ta lần lượt thử tách một học sinh ra khỏi K và kiểm tra xem nhóm K sau khi thử tách học sinh đó thì có thể rủ được cả lớp đi chơi hay không bằng cách áp dụng thuật toán câu 2. Nếu K vẫn rủ được Calop thì chứng tỏ học sinh vừa tách không phải là thành viên của nhóm nòng cốt của lớp. Nếu ngược lại, ta thực sự tách học sinh đó ra khỏi tập K.

Kết thúc quá trình thử đó với mọi học sinh trong lớp thì các học sinh còn lại trong K chính là nhóm nòng cốt cần tìm.

http://tieulun.hopto.org

```
Calop:= [hsd..hsc] ; K:= Calop;
For hs:=hsd to hsc do
Begin
  X:= K - [hs];
  áp dụng thuật toán câu 2 để tính Y từ X
  If Y = Calop Then K:=K-[hs];
End;
```

IV. HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP CHƯƠNG 4

Bài 4.3

Gợi ý:

Một điểm được định nghĩa là một biến bản ghi:

Diem = Record x,y: integer; end;

Về thuật toán:

Xét đoạn thẳng d đi qua 2 điểm A(x1,y1), B(x2,y2) và điểm M(x,y). Phương trình đoạn thẳng d có dạng:

(x-x1)/(x1-x2) =(y-y1)/(y1-y2)

Để phòng mẫu số bằng 0 ta viết thành dạng:

(x-x1)*(y1-y2)-(y-y1)*(x1-x2)=0

Đặt vế trái là f(M).

Nhận xét (*):

+ Nếu f(M) = 0 thì M nằm trên đoạn thẳng AB.

+ Nếu f(M) > 0 thì M nằm về một bên của đoạn AB và nếu f(M) < 0 thì điểm M nằm về phía bên kia của đoạn AB.

Để tổng quát trong việc xét vị trí tương đối của điểm M(x,y) so với đoạn thẳng AB, với A(x1,y1), B(x2,y2) cố định, ta có thể viết f(M) thành f(M,A,B).

Từ đó, xét 2 đoạn thẳng: đoạn AB =[(x1,y1),(x2,y2)] và đoạn CD = [(x3,y3),(x4,y4)]. Theo nhận xét (*) ta có nhận xét *(**)* sau đây:

+ Nếu tích P1= f(A,C,D)*f(B,C,D) có dấu âm thì chứng tỏ 2 điểm A và B nằm về 2 phía của đoạn thẳng CD. Còn nếu tích P1 có dấu dương thì 2 điểm A và B nằm về cùng một phía so với đoạn thẳng CD. Tất nhiên nếu P1 bằng 0 thì một trong 2 điểm A và/hoặc B rơi trên đoạn thẳng CD.

http://tieulun.hopto.org

+ Tương tự như thế ta có thể biết được vị trí tương đối của 2 điểm C và D so với đoạn thẳng AB bằng cách xét tích P2=f(C,A,B)*f(D,A,B) mang dấu gì.

+ Cuối cùng ta nhận thấy nếu tích P1 và P2 đều mang dấu âm thì chứng tỏ 2 đoạn thẳng AB và CD cắt nhau.

Tổng quát: Nếu có hàm logic:

g(A,B,C,D)=(P1<=0)and(P2<=0)

thì giá trị đúng của hàm g sẽ khẳng định 2 đoạn thẳng AB và CD có cắt nhau.

Bây giờ ta có thể sử dụng hàm g để giải quyết việc xét đường gấp khúc đi qua n điểm A_1, A_2,..., A_n có tự cắt hay không, n>=3.

Với mỗi đoạn A_iA_{i+1} (i=3,...,n-1) ta sẽ xem nó cắt các đoạn trước đó A_jA_{j+1} hay không (j=1,..,i-2) thông qua việc xét giá trị của hàm $g(A_i,A_{i+1},A_j,A_{j+1})$

Trong chương trình nên xây dựng các hàm sau đây:

```
1) Function f(M,A,B:Diem): Real;
begin
   f:= (M.x-A.x)*(A.y-B.y)-(M.y-A.y)*(A.x-B.x)
end;
```

```
2) Function g(M,N,A,B:Diem): Boolean;
Var P1,P2: real;
Begin
 P1:=f(M,A,B)*f(N,A,B);
 P2:=f(A,M,N)*f(B,M,N);
 Catnhau:= (P1<=0) and (P2<=0);
End;
```

```
3) Function GhapKhucTuCat( A: Arr): Boolean;
Var i,j: integer;
Begin
For i:=3 to n-1 do
For j:=1 to i-2 do
If g(A[i],A[i+1],A[j],A[j+1]) then
Begin
```


http://tieulun.hopto.org

```
    GhapKhucTuCat:= True;
    Exit;
  End;
 GhapKhucTuCat:= False;
 End;
```

V. HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP CHƯƠNG 5

Bài 5.8

Về tổ chức dữ liệu:

Nên khai báo kiểu Str30 = string[30] để truyền tham số kiểu string[30] vào thủ tục.

```
  Str30 = string[30];
  Hocsinh = Record
   ht: Str30;
   ns: integer;
   lop: string[10];
   xlhk, xlhl: string[3];
 End;
```

Vài điểm lưu ý trong chương trình:

Chương trình chính chỉ tổ chức việc trình bày menu chính. Mỗi mục của menu này sẽ gọi các thủ tục thực hiện các nhiệm vụ tương ứng.

```
 Write('Ban chon muc nao: '); Readln(chon);
  Case chon of
  1: Append;
  2: List;
  3: Find;
  End;
 Until chon=4;
```

Trong 3 thủ tục trên, nên sử dụng thêm 3 modul cơ bản sau đây để tiện sử dụng trong các thủ tục chính:

- Procedure Input(Var hs: Hocsinh):

http://tieulun.hopto.org

Làm nhiệm vụ nhập thông tin đầy đủ cho một học sinh hs.

• Procedure Output(hs: Hocsinh):

Làm nhiệm vụ in lên màn hình thông tin đầy đủ của một học sinh hs.

• Function NameOf(Fullname: str30): Str30:

Là hàm cho giá trị bằng từ cuối cùng của chuỗi ký tự Fullname (hoặc là chính chuỗi đó trong trường hợp chuỗi này chỉ có một từ), đó chính là phần tên trong tên đầy đủ của một học sinh. Lưu ý rằng chương trình chỉ yêu cầu tìm theo tên, do đó không nhất thiết phải nhập tên đầy đủ khi tìm kiếm.

Chương trình

```
Program QLHS;
Uses Crt;
const fname = 'QLHS.DAT';
Type
 Str30 = string[30];
 Hocsinh = Record
  ht: Str30;
  ns: integer;
  lop: string[10];
  xlhk, xlhl: string[3];
End;

Var f: FILE OF Hocsinh;
   chon: byte;
(*-------------------------------*)
Procedure Input(Var hs: Hocsinh);
Begin
 With hs do
 Begin
  Write('Ho va ten: '); Readln(ht);
  if ht='' then exit;
  Write('Nam sinh: '); Readln(ns);
```


http://tieulun.hopto.org

```
    Write('Lop: '); Readln(lop);
    Write('XL Hanh kiem: '); Readln(xlhk);
    Write('XL Hoc luc: '); Readln(xlhl);
  End;
End;
(*-------------------------------*)
Procedure Output(hs: Hocsinh);
Begin
 With hs do
 Begin
  Writeln('Ho va ten: ',ht);
  Writeln('Nam sinh: ',ns);
  Writeln('Lop: ',lop);
  Writeln('XL Hanh kiem: ',xlhk);
  Writeln('XL Hoc luc: ',xlhl);
  Writeln('----------------------');
 End;
 Readln;
End;
(*-------------------------------*)
Procedure Append;
var k: integer;
   hs: Hocsinh;
Begin
 Clrscr;
 k:= FileSize(f);
 If k=0 Then Writeln('Chua co hoc sinh nao trong tep')
 Else Writeln('Da co ',k,' hoc sinh ');
 seek(f,k);
 Input(hs);
 if hs.ht <>'' then Write(f,hs);
 Writeln('Enter de tro ve menu chuong trinh');
```


http://tieulun.hopto.org

```
 Readln;
End;
(*-------------------------------*)
Procedure List;
Var i: integer;
   hs: Hocsinh;
Begin
 Clrscr;
 Reset(f);
 For i:=1 to FileSize(f) do
 Begin
  Read(f,hs);
  Output(hs);
 End;
End;
(*-------------------------------*)
Function NameOf(Fullname: str30): Str30;
Var i: byte;
Begin
 i:=length(Fullname);
 while (i>0) and (Fullname[i]<>#32) do dec(i);
 if i=0 then NameOf:= Fullname
 Else NameOf:= Copy(Fullname, i + 1, length(Fullname));
End;
(*-------------------------------*)
Procedure Find;
Var i: integer;
   hs: Hocsinh;
   ten: string[6];
   Found: Boolean;
Begin
 Clrscr;
```


http://tieulun.hopto.org

```
Write('Nhap ten hoc sinh can tim: '); Readln(ten);
If ten='' then Exit;
Found:= False;
Reset(f);
For i:=1 to FileSize(f) do
Begin
 Read(f,hs);
 If NameOf(hs.ht) = ten then
 Begin Writeln('Tim thay hoc sinh thu ',i);
    Output(hs);
    Found:= True;
  End;
 End;
 If Not Found Then Writeln('Khong co hoc sinh ten la ',ten);
 Writeln('Enter tro ve menu chuong trinh');
 Readln;
End;
(*-------------------------------*)
BEGIN
Assign(f, fname);
{$I-}
Reset(f);
{I-}
If IOResult > 0 Then Rewrite(f);

Repeat
 Clrscr;
 Writeln('Menu chuong trinh');
 Writeln('1. Nhap hoc sinh moi');
 Writeln('2. Xem danh sach hoc sinh');
 Writeln('3. Tim theo ten hoc sinh');
 Writeln('4. Thoat khoi chuong trinh');
```


http://tieulun.hopto.org

```
  Write('Ban chon muc nao: '); Readln(chon);
  Case chon of
  1: Append;
  2: List;
  3: Find;
  End;
Until chon=4;
Close(f);
END.
```

Bài 5.9

Vài điểm gợi ý:

Việc tổ chức dữ liệu và tổ chức chương trình chính cùng với các mục 1.2.3 của menu giống như bài tập 5.8.

```
Write('Ban chon muc nao: '); Readln(chon);
  Case chon of
  1: Append;
  2: List;
  3: FindAuthor;
  4: FindBooks;
End;
```

Trọng tâm cần bàn tới là giải quyết phép so sánh tương đối 2 xâu ký tự - mục 4 của menu. Thủ tục FindBooks có nội dung chính sau đây:

```
Reset(f);
  For i:=1 to FileSize(f) do
  Begin
    Read(f,s);
    If Compare(ten,s.tens) then
    Begin Writeln('Tim thay quyen sach o vi tri ',i);
        Output(s);
        Found:= True;
  End;
End;
If Not Found Then Writeln('Khong co quyen sach nao nhu vay.');
```


http://tieulun.hopto.org

Hàm Compare(x,s) là hàm so sánh xấp xỉ tên gần đúng quyển sách là x có trong tên chính xác quyển sách s hay không.

Trước hết phải xoá hết các dấu cách hai và các dấu cách vô nghĩa trong x.

```
while x[1]=#32 do delete(x,1,1);
while x[length(x)]=#32 do delete(x,length(x),1);
repeat k:=pos(#32+#32,x); if k>0 then delete(x,k,1);
Until k=0;
```

Tiếp theo, nếu x là một từ thì x phải là một từ nào đó trong tên chính xác của cuốn sách s. Hàm Pos_word(x,s) dưới đây cho biết từ x có trùng với một từ nào đó trong s hay không. Ta tạm thời chưa xét hàm Pos_word(x,s) để quan tâm tiếp việc biện luận đang diễn ra ở hàm Compare(x,s).

```
If pos(#32,x)=0 then
 begin
  Compare:= pos_word(x,s);
  exit;
 end;
```

Trong trường hợp x gồm nhiều từ thì tập tất cả các từ của tên sách gần đúng x, từng từ một phải trùng với một từ nào đó của tên sách chính xác s:

```
got:= true;
 Repeat
 k:= pos(#32,x);
 if k>0 then
 begin
   w:= copy(x,1,k-1);
   got:= got and pos_word(w,s);
   delete(x,1,k);
   {if pos_word(w,s) then Writeln('Co ',w,' trong ',s);}
end;
Until k=0;
Compare:= got and pos_word(x,s);
```


http://tieulun.hopto.org

Như vậy hàm Compare(x,s) đã được xem xét đầy đủ.

Xét tiếp hàm Pos_word(x,s)

Nếu s chỉ gồm một từ thì việc so sánh diễn ra ở 2 từ và thực hiện bởi phép so sánh bằng nhau thông thường:

```
If pos(#32,s) = 0 then
 Begin
  Pos_Word:= (x = s);
  Exit;
 End;
```

Nếu s gồm nhiều từ thì Pos_word được thoả (bằng true) khi một từ nào đó của s bằng từ x.

```
Exist:= False;
Repeat
 i:= pos(#32,s);
 if i>0 then
 begin
   ws:= copy(s,1,i-1);
   Exist:= Exist Or (x=ws);
   delete(s,1,i);
end;
Until i=0;
Pos_Word:= Exist Or (x=s);
```

Chương trình:

```
Program QLSACH;
Uses Crt;
const fname = 'QLSACH.DAT';
Type
 Str30 = string[30];
 SACH = Record
  tens: Str30;
```


http://tieulun.hopto.org

```
    tacgia: str30;
    namxb: integer;
    nhaxb: str30;
End;
Var f: FILE OF SACH;
    chon: byte;
(*-------------------------------*)
Procedure Input(Var s: SACH);
Begin
 With s do
 Begin
  Write('Ten sach: '); Readln(tens);
  if s.tens='' then exit;
  Write('Tac gia: '); Readln(tacgia);
  Write('Nam xuat ban: '); Readln(namxb);
  Write('Nha xuat ban: '); Readln(nhaxb);
 End;
End;
(*-------------------------------*)
Procedure Output(s: SACH);
Begin
 With s do
 Begin
  Writeln('Ten sach: ',tens);
  Writeln('Tac gia: ',tacgia);
  Writeln('Nam xuat ban: ',namxb);
  Writeln('Nha xuat ban: ',nhaxb);
  Writeln('----------------------');
 End;
 Readln;
End;
(*-------------------------------*)
```


http://tieulun.hopto.org

```
Procedure Append;
var k: integer;
  s: SACH;
Begin
 Clrscr;
 k:= FileSize(f);
 If k=0 Then Writeln('Chua co quyen sach nao trong tep')
 Else Writeln('Da co ',k,' quyen sach');
 seek(f,k);
 Input(s);
 if s.tacgia <>'' then Write(f,s);
 Writeln('Enter de tro ve menu chuong trinh');
 Readln;
End;
(*-------------------------------*)
Procedure List;
Var i: integer;
  s: SACH;
Begin
 Clrscr;
 Reset(f);
 For i:=1 to FileSize(f) do
 Begin
  Read(f,s);
  Output(s);
 End;
End;
(*-------------------------------*)
Function NameOf(Fullname: str30): Str30;
Var i: byte;
Begin
 i:=length(Fullname);
```


http://tieulun.hopto.org

```
 while (i>0) and (Fullname[i]<>#32) do dec(i);
 if i=0 then NameOf:= Fullname
 Else NameOf:= Copy(Fullname, i + 1, length(Fullname));
End;
(*-------------------------------*)
Procedure FindAuthor;
Var i: integer;
  s: SACH;
  ten: string[6];
  Found: Boolean;
Begin
 Clrscr;
 Write('Nhap ten tac gia can tim: '); Readln(ten);
 If ten='' then Exit;
 Found:= False;
 Reset(f);
 For i:=1 to FileSize(f) do
Begin
 Read(f,s);
 If NameOf(s.tacgia) = ten then
 Begin Writeln('Tim thay tac gia thu ',i);
    Output(s);
    Found:= True;
 End;
End;
If Not Found Then Writeln('Khong co tac gia la ',ten);
Writeln('Enter tro ve menu chuong trinh');
Readln;
End;
(*-------------------------------*)
Function Pos_Word(x , s: str30): Boolean;
Var i: byte; ws: str30; exist: Boolean;
```


http://tieulun.hopto.org

```
    ws:= copy(s,1,i-1);
    Exist:= Exist Or (x=ws);
    delete(s,1,i);
  end;
  Until i=0;
  Pos_Word:= Exist Or (x=s);
End;
(*-----------------------------------*)
Function Compare(x,s: str30): Boolean;
Var k: byte; got: Boolean; w: Str30;
Begin
 while x[1]=#32 do delete(x,1,1);
 while x[length(x)]=#32 do delete(x,length(x),1);
 repeat
  k:=pos(#32+#32,x);
  if k>0 then delete(x,k,1);
Until k=0;
If pos(#32,x)=0 then
begin
 Compare:= pos_word(x,s);
 exit;
```


http://tieulun.hopto.org

```
Begin
 If pos(#32,s) = 0 then
 Begin
  Pos_Word:= (x = s);
  Exit;
 End;
 Exist:= False;
 Repeat
 i:= pos(#32,s);
 if i>0 then
 begin
   ws:= copy(s,1,i-1);
   Exist:= Exist Or (x=ws);
   delete(s,1,i);
 end;
 Until i=0;
 Pos_Word:= Exist Or (x=s);
End;
(*-------------------------------*)
Function Compare(x,s: str30): Boolean;
Var k: byte; got: Boolean; w: Str30;
Begin
 while x[1]=#32 do delete(x,1,1);
 while x[length(x)]=#32 do delete(x,length(x),1);
 repeat
  k:=pos(#32+#32,x);
  if k>0 then delete(x,k,1);
 Until k=0;
 If pos(#32,x)=0 then
 begin
  Compare:= pos_word(x,s);
  exit;
```


http://tieulun.hopto.org

```
    end;
    got:= true;
    Repeat
    k:= pos(#32,x);
    if k>0 then
    begin
      w:= copy(x,1,k-1);
      got:= got and pos_word(w,s);
      delete(x,1,k);
      {if pos_word(w,s) then Writeln('Co ',w,' trong ',s);}
    end;
    Until k=0;
    Compare:= got and pos_word(x,s);
End;
(*-------------------------------*)
Procedure FindBooks;
Var i: integer;
  s: SACH;
  ten: str30;
  Found: Boolean;
Begin
 Clrscr;
 Write('Nhap ten sach gan dung: '); Readln(ten);
 If ten='' then Exit;
 Found:= False;
 Reset(f);
 For i:=1 to FileSize(f) do
 Begin
  Read(f,s);
  If Compare(ten,s.tens) then
  Begin Writeln('Tim thay quyen sach o vi tri ',i);
     Output(s); Found:= True;
```


http://tieulun.hopto.org

```
  End;
 End;
 If Not Found Then Writeln('Khong co quyen sach nao nhu vay.');
 Writeln('Enter tro ve menu chuong trinh'); Readln;
End;
(*-------------------------------*)
BEGIN
Assign(f, fname);
{$I-}
Reset(f);
{I-}
If IOResult > 0 Then Rewrite(f);
Repeat
 Clrscr;
 Writeln('Menu chuong trinh');
 Writeln('1. Nhap them mot quyen sach');
 Writeln('2. Xem danh muc sach');
 Writeln('3. Tim sach theo ten tac gia');
 Writeln('4. Tim sach theo ten gan dung cua sach');
 Writeln('5. Thoat khoi chuong trinh');
 Write('Ban chon muc nao: '); Readln(chon);
 Case chon of
 1: Append;
 2: List;
 3: FindAuthor;
 4: FindBooks;
 End;
Until chon=5;
Close(f);
END.
```


http://tieulun.hopto.org

VI. HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP PHẦN PHỤ LỤC

Bài PL2.1

Gợi ý

Ta gọi chia(m,n) là số lượng cách chia m gói quà cho n người.

Nếu số người n=0 thì số cách chia =0, vì không có cách chia cho 0 người.

Nếu số quà m =0 thì số cách chia =1, nghĩa là mỗi người nhận được 0 quà.

Nếu m<n thì số người ít hơn số quà và hơn nữa người trước không nhận ít quà hơn người sau nên từ người thứ m+1 sẽ không nhận được quà. Trong trường hợp này số cách chia quà bằng số cách chia m quà cho m người. Chia:=chia(m,m). Ta xét tiếp trường hợp m=n dưới đây.

Nếu m>=n thì số cách chia m quà cho n người bằng tổng số cách chia của 2 nhóm S1 và S2 sau đây:

Nhóm 1: Gồm các cách chia mà người thứ n không nhận được quà, nghĩa là chỉ chia quà cho n-1 người đầu tiên:

S1=chia(m,n-1)

Nhóm 2: Gồm các cách chia mà người thứ n nhận được ít nhất một gói quà. Do đó ai cũng nhận được quà, vì vậy ta có thể bớt mỗi người 1 gói quà, cho họ lĩnh sau. Số m-n quà còn lại sẽ chia cho n người:

S2:=chia(m-n,n)

Tóm lại, khi m>=n ta có:

chia := chia(m,n-1) + chia(m-n,n).

Và, ta có thủ tục đệ quy sau đây:

```
Function Socach(m,n:byte):Integer;
Begin
 If n = 0 Then Socach:=0
 Else If m=0 Then Socach:=1
  Else If m<n Then Socach: = Socach(m,m)
       Else Socach:= Socach(m,n-1) + Socach(m-n,n);
End;
```

Bài PL2.2

Gợi ý

Bài này có thể làm tương tự như bài phân tích số tự nhiên N.

http://tieulun.hopto.org

Ta có thủ tục đệ quy sau đây:

```
Procedure Tach(n:integer);
Var k: integer;
Begin
 { Tim so hang fibonaci thu k gan sat voi n nhat}
  k:=0; while u[k]<=n do inc(k); dec(k);
  x[k]:=1;
 If n>u[k] then Tach(n-u[k]);
End;
```

Trong đó, u[1..100] là mảng chứa dãy fibonaci.

Bài PL2.3

Gợi ý

Ta gọi Tim(L,R,n) là thủ tục đệ quy xét trên đoạn thứ tự là n từ toạ độ trái L đến toạ độ phải R. Thứ tự n theo quy định mô tả ở đầu bài.

Đoạn thẳng trước khi chia có thứ tự là 1. Nghĩa là ở chương trình chính gọi Tim(0,k,1).

Số m quy định số lần chia các đoạn thẳng còn lại, đồng thời cũng là số lần gọi đệ quy thủ tục Tim(L,R,n). Như vậy có một biến toàn cục để đếm số lần chia, ta gọi biến đó là slchia.

Thủ tục Tim(L,R,n) thực hiện các công việc sau đây:

Số hiệu đoạn	n		
Số hiệu đoạn	2(n-1)+1		2(n-1)+2
L	U	V	R

Tính toạ độ (thực chất là hoành độ) 2 đầu đoạn thẳng ở giữa:

U:=L+(R-L)/3

V:=L+2(R-L)/3

Tăng số lần chia inc(slchia);

Ta có 2 trường hợp:

Trường hợp 1: Nếu số lần chia <m thì ta có:

Nếu x/y <= U thì: Ta xét đoạn (L,U) với thứ tự đoạn tương ứng là 2(n-1)+1 giống như xét đoạn (L,R) nói trên với thứ tự đoạn là n.

Nếu x/y >=V thì: Ta xét đoạn (U,R) với thứ tự đoạn tương ứng là 2(n-1)+2 giống như xét đoạn (L,R) nói trên với thứ tự đoạn là n.

http://tieulun.hopto.org

Ngược lại, quá trình đệ quy chấm dứt và chắc chắn x/y không thuộc đoạn nào dù tiếp tục chia nữa.

Trường hợp 2: Nếu số lần chia đã = m, ta có:

Nếu x/y <= U thì: Phân số x/y thuộc đoạn 2(n-1)+1.

Nếu x/y >=V thì: Phân số x/y thuộc đoạn 2(n-1)+2.

Nếu không phải 2 trường hợp trên thì phân số x/y không thuộc đoạn nào ở lần chia cuối cùng này. Cụ thể nó thuộc đoạn chưa bị gạch bỏ ở lần chia trước đó, nhưng lại nằm đúng phần bị gạch bỏ ở lần chia cuối cùng.

Vậy ta có thủ tục đệ quy sau đây:

```
Procedure Tim(l,r:real; n:integer);
var u,v: real;
Begin
 u:=l+(r-l)/3;
 v:=l+2*(r-l)/3;
 inc(slchia);
 if slchia<m then
begin
 if x/y <= u then Tim(l,u,2*(n-1)+1)
 else
 if x/y >= v then Tim(v,r,2*(n-1)+2)
 else
 begin writeln('Khong thuoc');Exit; End;
end
else
begin
 if (x/y<=u) then writeln(2*(n-1)+1)
 else
 if x/y >= v then Writeln(2*(n-1)+2)
 else writeln(f,'Khong thuoc');
 end;
End;
```

Ở chương trình chính, ta khởi tạo slchia=0 và gọi Tim(0,K,1);

http://tieulun.hopto.org

TÀI LIỆU THAM KHẢO

1. Larry Nyhoff Sanford Leestma - ***Lập trình nâng cao bằng Pascal với các cấu trúc dữ liệu (tập 1& tập 2)*** - Người dịch: Lê Minh Trung - Công ty liên doanh tư vấn và dịch vụ khoa học kỹ thuật - Nhà in số 3 - Công ty Scitec - 1997.
2. Kenneeth H.Rosen - ***Toán học rời rạc ứng dụng trong tin học*** - Người dịch Phạm Văn Thiều, Đặng Hữu Thịnh - Khoa Công nghệ - Đại học Khoa học Tự nhiên - Đại học Quốc gia Hà Nội - Nhà xuất bản Khoa học kỹ thuật - 1997.
3. Robert Sedgewich - ***Cẩm nang thuật toán (tập 1 và tập 2)*** - Người dịch: Trần Đan Thư, Vũ Mạnh Tường, Dương Vũ Diệu Trà, Nguyễn Tiến Huy - Nhà xuất bản Khoa học kỹ thuật - tái bản 2001.
4. Phạm Văn Ất - Đại học Giao thông vận tải - ***Turro Pascal 5 & 6 (Giáo trình cơ sở và nâng cao kỹ thuật lập trình hướng đối tượng)*** - Nhà xuất bản Giáo dục - 1993.
5. Nguyễn Xuân Huy - Viện Công nghệ thông tin - ***Thuật toán*** - Nhà xuất bản Thống kê - 1994.

http://tieulun.hopto.org

MỤC LỤC

Lời giới thiệu 3

Lời nói đầu 5

Chương 1. CÁC VẤN ĐỀ CƠ BẢN CỦA THUẬT TOÁN 7

I. Thuật toán và một số tính chất cơ bản 7

1. Khái niệm thuật toán 7

2. Một số tính chất cơ bản của thuật toán 8

3. Câu hỏi và bài tập 10

II. Các cấu trúc điều khiển trong thuật toán 10

1. Các hình thức mô tả thuật toán 10

2. Các cấu trúc điều khiển trong thuật toán 14

3. Câu hỏi và bài tập 25

III. Giới thiệu các ngôn ngữ lập trình 26

1. Khái niệm 26

2. Phân loại (ngôn ngữ bậc thấp. ngôn ngữ bậc cao) 26

3. Tóm tắt về các lớp ngôn ngữ (đọc thêm) 27

4. Thông dịch và biên dịch (đọc thêm) 29

5. Câu hỏi và bài tập 30

Chương 2. CÁC KIỂU DỮ LIỆU CƠ BẢN VÀ CÁC CẤU TRÚC ĐIỀU KHIỂN .. 32

I. Các thao tác cơ bản trong Turbo Pascal 32

1. Các file của Turbo Pascal 32

2. Vào/ ra Turbo Pascal 32

3. Tóm tắt các phím tắt để quản lý file 33

4. Tóm tắt các thao tác soạn thảo 33

5. Các bước thực hành 33

II. Giới thiệu chung về ngôn ngữ lập trình Pascal 34

1. Các phần tử cơ sở của ngôn ngữ lập trình Pascal 34

http://tieulun.hopto.org

2. Kiểu dữ liệu, hằng, biến, biểu thức và câu lệnh gán..... 35
3. Các lệnh vào/ra..... 38
4. Cấu trúc cơ bản của một chương trình Pascal..... 40
5. Câu hỏi và bài tập..... 44
III. Các dữ liệu cơ bản..... 44
1. Đặt vấn đề..... 44
2. Tổng quan phân loại..... 45
3. Các kiểu dữ liệu đơn giản chuẩn..... 45
4. Câu hỏi và bài tập..... 49
IV. Các cấu trúc điều khiển trong ngôn ngữ lập trình Pascal..... 50
1. Cấu trúc tuần tự..... 50
2. Cấu trúc rẽ nhánh..... 50
3. Cấu trúc lặp với số lần biết trước..... 55
4. Cấu trúc lặp với số lần không biết trước..... 63
5. Câu hỏi và bài tập..... 67
Chương 3. CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC CƠ BẢN..... 70
I. Cấu trúc dữ liệu kiểu mảng một chiều..... 70
1. Định nghĩa, khai báo mảng một chiều..... 70
2. Vào/ ra mảng một chiều..... 71
3. Duyệt mảng và kỹ thuật 3 bước trong lập trình..... 73
4. Một số thuật toán tìm kiếm đơn giản..... 75
5. Một số thuật toán sắp xếp đơn giản..... 78
6. Câu hỏi và bài tập..... 88
II. Cấu trúc dữ liệu kiểu mảng hai chiều..... 90
1. Định nghĩa, khai báo mảng hai chiều..... 90
2. Vào/ ra mảng hai chiều..... 92
3. Duyệt mảng hai chiều và kỹ thuật 3 bước trong lập trình..... 93
4. Câu hỏi và bài tập..... 95

http://tieulun.hopto.org

III. Dữ kiểu liệu kiểu xâu ký tự..... 98
1. Định nghĩa, khai báo xâu ký tự..... 98
2. Các phép toán trên xâu ký tự..... 98
3. Các hàm và thủ tục trên xâu ký tự..... 99
4. Một số thuật toán - bài toán xử lý xâu ký tự..... 101
5. Câu hỏi và bài tập..... 109
IV. Dữ liệu kiểu bản ghi..... 110
1. Định nghĩa, khai báo bản ghi..... 110
2. Cách truy nhập các trường của biến bản ghi..... 111
3. Các phép toán..... 113
4. Ví dụ bài tập về bản ghi..... 113
5. Câu hỏi và bài tập..... 118
V. Dữ liệu kiểu đoạn con, liệt kê và tập hợp..... 119
1. Dữ liệu kiểu đoạn con và kiểu liệt kê..... 120
2. Dữ liệu kiểu tập hợp..... 121
3. Câu lệnh chọn lựa..... 124
4. Câu hỏi và bài tập..... 128
Chương 4. CHƯƠNG TRÌNH CON..... 130
I. Chương trình có chương trình con..... 130
1. Khái niệm chương trình con..... 130
2. Cấu trúc một chương trình có chương trình con..... 132
3. So sánh thủ tục và hàm..... 133
4. Câu hỏi và bài tập..... 135
II. Tham biến và tham trị, biến toàn cục và biến địa phương..... 136
1. Biến toàn cục và biến địa phương: So sánh, cách sử dụng..... 136
2. Tham biến và tham trị: So sánh, cách sử dụng..... 140
3. Tham chiếu (đọc thêm)..... 143
4. Câu hỏi và bài tập..... 143

http://tieulun.hopto.org

Chương 5. DỮ LIỆU KIỂU TỆP (FILE)..... 146
I. Tệp văn bản..... 147
1. Tệp văn bản ngầm định Input và Output..... 147
2. Cách sử dụng tệp văn bản..... 148
3. Một số kỹ năng sử dụng tệp văn bản..... 155
4. Câu hỏi và bài tập..... 165
II. Tệp định kiểu..... 167
1. Định nghĩa tệp định kiểu..... 167
2. Khai báo biến tệp định kiểu..... 168
3. Cách sử dụng tệp định kiểu..... 169
4. Ví dụ việc sử dụng tệp định kiểu..... 171
5. Câu hỏi và bài tập..... 175
Phụ lục 1: ĐỘ PHỨC TẠP CỦA THUẬT TOÁN..... 177
I. Tính hiệu quả của thuật toán..... 177
II. Độ phức tạp của thuật toán..... 179
1. Khái niệm về độ tăng của hàm..... 179
2. Độ phức tạp thuật toán..... 180
3. Câu hỏi và bài tập..... 187
Phụ lục 2: THUẬT TOÁN ĐỆ QUY VÀ CHƯƠNG TRÌNH CON ĐỆ QUY..... 189
I. Các khái niệm về đệ quy..... 189
II. Chương trình con đệ quy..... 191
1. Cách viết một chương trình con đệ quy..... 191
2. Bản chất của chương trình con đệ quy..... 192
III. Phân tích quy nạp trong đệ quy..... 195
1. Bài toán Tháp Hà Nội..... 196
2. Bài toán chia phần..... 198
3. Bài toán phân tích số..... 199
4. Bài toán tìm các hoán vị..... 201
5. Câu hỏi và bài tập..... 205

http://tieulun.hopto.org

HƯỚNG DẪN GIẢI MỘT SỐ BÀI TẬP..... 207

I. Hướng dẫn giải một số bài tập chương 1..... 207

II. Hướng dẫn giải một số bài tập chương 2..... 209

III. Hướng dẫn giải một số bài tập chương 3..... 214

IV. Hướng dẫn giải một số bài tập chương 4..... 226

V. Hướng dẫn giải một số bài tập chương 5..... 228

VI. Hướng dẫn giải một số bài tập phần phụ lục..... 242

Tài liệu tham khảo..... 245

http://tieulun.hopto.org

NHÀ XUẤT BẢN HÀ NỘI
4 - TỐNG DUY TÂN, QUẬN HOÀN KIẾM, HÀ NỘI
ĐT: (04) 8252916, 8257063 - FAX: (04) 8257063

GIÁO TRÌNH

THUẬT TOÁN VÀ KỸ THUẬT LẬP TRÌNH PASCAL

NHÀ XUẤT BẢN HÀ NỘI - 2005

Chịu trách nhiệm xuất bản:
NGUYỄN KHẮC OÁNH
Biên tập:
TRƯƠNG ĐỨC HÙNG
Bìa:
TRẦN QUANG
Trình bày - Kỹ thuật vi tính:
THU HIỀN
Sửa bản in:
PHẠM THU TRANG

http://tieulun.hopto.org

In 2.060 cuốn, khổ 17x24cm tại Công ty In Khoa học kỹ thuật
101A Nguyễn Khuyến - Đống Đa - Hà Nội.
Số in:74. Giấy phép xuất bản: 198GT/290 CXB cấp ngày 14/3/2005.
In xong và nộp lưu chiểu tháng 4 năm 2005.

http://tieulun.hopto.org

BỘ GIÁO TRÌNH XUẤT BẢN NĂM 2005
KHỐI TRƯỜNG TRUNG HỌC KINH TẾ KỸ THUẬT TIN HỌC

1. THUẬT TOÁN LẬP TRÌNH
2. ĐÁNH MÁY VI TÍNH
3. SOẠN THẢO VÀ ĐÁNH MÁY VĂN BẢN
4. NGHIỆP VỤ THƯ KÝ
5. KẾ TOÁN MÁY
6. MARKETING
7. NGÔN NGỮ LẬP TRÌNH C
8. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI (C++)
9. BẢNG TÍNH ĐIỆN TỬ (EXCEL)
10. VISUAL BASIC
11. CẤU TRÚC MÁY TÍNH
12. GIAO TIẾP
13. ACCESS
14. MẠNG MÁY TÍNH
15. THIẾT KẾ WEB
16. BẢO TRÌ PC
17. HỆ ĐIỀU HÀNH
18. CƠ SỞ DỮ LIỆU
19. PHÂN TÍCH THIẾT KẾ HỆ THỐNG
20. KỸ THUẬT SỐ
21. PHOTOSHOP
22. CAD/CAM

¥509 194

GT THUAT TOAN VA KY THUAT LT..
10152317
08/04/5
33.000 d/C

Giá: 33.000d

http://tieulun.hopto.org