BỘ GIÁO DỤC VÀ ĐÀO TẠO

TIN HỌC

TIN HỌC 11

NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

TIN HỌC
11

BỘ GIÁO DỤC VÀ ĐÀO TẠO

HỒ SĨ ĐÀM (Chủ biên) - HỒ CẨM HÀ - TRẦN ĐỖ HÙNG
NGUYỄN ĐỨC NGHĨA - NGUYỄN THANH TÙNG - NGÔ ÁNH TUYẾT

TIN HỌC 11

(Tái bản lần thứ mười ba)

NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

Hãy bảo quản, giữ gìn sách giáo khoa để dành tặng cho các em học sinh lớp sau !

Chương 1

MỘT SỐ KHÁI NIỆM VỀ LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

§1. KHÁI NIỆM LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

Như ta biết, mọi bài toán có thuật toán đều có thể giải được trên máy tính điện tử. Khi giải bài toán trên máy tính điện tử, sau các bước xác định bài toán và xây dựng hoặc lựa chọn thuật toán khả thi là bước lập trình.

Lập trình là sử dụng cấu trúc dữ liệu và các câu lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán. Chương trình viết bằng ngôn ngữ lập trình bậc cao nói chung không phụ thuộc vào loại máy, nghĩa là một chương trình có thể thực hiện trên nhiều loại máy tính khác nhau. Chương trình viết bằng ngôn ngữ máy có thể được nạp trực tiếp vào bộ nhớ và thực hiện ngay, còn chương trình viết bằng ngôn ngữ lập trình bậc cao phải được chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện được.

Chương trình đặc biệt có chức năng chuyển đổi chương trình được viết bằng ngôn ngữ lập trình bậc cao thành chương trình thực hiện được trên máy tính được gọi là *chương trình dịch*.

Chương trình dịch nhận đầu vào là chương trình viết bằng ngôn ngữ lập trình bậc cao (chương trình nguồn), thực hiện chuyển đổi sang ngôn ngữ máy (chương trình đích).

Xét ví dụ, bạn chỉ biết tiếng Việt nhưng cần giới thiệu về trường của mình cho đoàn khách đến từ nước Mĩ, chỉ biết tiếng Anh. Có hai cách để bạn thực hiện điều này.

Cách thứ nhất: Bạn nói bằng tiếng Việt và người phiên dịch giúp bạn dịch sang tiếng Anh. Sau mỗi câu hoặc một vài câu giới thiệu trọn một ý, người phiên dịch dịch sang tiếng Anh cho đoàn khách. Sau đó, bạn lại giới thiệu tiếp và người phiên dịch lại dịch tiếp. Việc giới thiệu của bạn và việc dịch của người phiên dịch luân phiên cho đến khi bạn kết thúc nội dung giới thiệu của mình. Cách dịch trực tiếp như vậy được gọi là *thông dịch*.

Cách thứ hai: Bạn soạn nội dung giới thiệu của mình ra giấy, người phiên dịch dịch toàn bộ nội dung đó sang tiếng Anh rồi đọc hoặc trao văn bản đã dịch cho đoàn khách đọc. Như vậy, việc dịch được thực hiện sau khi nội dung

giới thiệu đã hoàn tất. Hai công việc đó được thực hiện trong hai khoảng thời gian độc lập, tách biệt nhau. Cách dịch như vậy được gọi là *biên dịch*.

Sau khi kết thúc, với cách thứ nhất không có một văn bản nào để lưu trữ, còn với cách thứ hai có hai bản giới thiệu về trường của bạn bằng tiếng Việt và bằng tiếng Anh có thể lưu trữ để dùng lại về sau.

Tương tự như vậy, chương trình dịch có hai loại là *thông dịch* và *biên dịch*.

a) Thông dịch

Thông dịch (interpreter) được thực hiện bằng cách lặp lại dãy các bước sau:

① Kiểm tra tính đúng đắn của câu lệnh tiếp theo trong chương trình nguồn;

② Chuyển đổi câu lệnh đó thành một hay nhiều câu lệnh tương ứng trong ngôn ngữ máy;

③ Thực hiện các câu lệnh vừa chuyển đổi được.

Như vậy, quá trình dịch và thực hiện các câu lệnh là luân phiên. Các chương trình thông dịch lần lượt dịch và thực hiện từng câu lệnh. Loại chương trình dịch này đặc biệt thích hợp cho môi trường đối thoại giữa người và hệ thống. Tuy nhiên, nếu một câu lệnh nào đó phải thực hiện bao nhiêu lần thì nó phải được dịch bấy nhiêu lần.

Các ngôn ngữ khai thác hệ quản trị cơ sở dữ liệu, ngôn ngữ đối thoại với hệ điều hành,... đều sử dụng trình thông dịch.

b) Biên dịch

Biên dịch (compiler) được thực hiện qua hai bước:

① Duyệt, phát hiện lỗi, kiểm tra tính đúng đắn của các câu lệnh trong chương trình nguồn;

② Dịch toàn bộ chương trình nguồn thành một chương trình đích có thể thực hiện trên máy và có thể lưu trữ để sử dụng lại khi cần thiết.

Như vậy, trong thông dịch, không có chương trình đích để lưu trữ, trong biên dịch cả chương trình nguồn và chương trình đích có thể lưu trữ lại để sử dụng về sau.

Thông thường, trong môi trường làm việc trên một ngôn ngữ lập trình cụ thể, ngoài chương trình dịch còn có một số thành phần có chức năng liên quan như biên soạn, lưu trữ, tìm kiếm, cho biết các kết quả trung gian,.... Ví dụ, Turbo Pascal 7.0, Free Pascal 1.2, Visual Pascal 2.1,... là các môi trường làm việc trên ngôn ngữ Pascal; Turbo C++, Visual C++,... là các môi trường làm việc trên ngôn ngữ C++.

Các môi trường lập trình khác nhau ở những dịch vụ mà nó cung cấp, đặc biệt là các dịch vụ nâng cấp, tăng cường các khả năng mới cho ngôn ngữ lập trình.

Bài đọc thêm 1

BẠN BIẾT GÌ VỀ CÁC NGÔN NGỮ LẬP TRÌNH?

Đã có hàng nghìn ngôn ngữ lập trình được thiết kế và mỗi năm lại có thêm nhiều ngôn ngữ lập trình mới xuất hiện. Các ngôn ngữ thường được nhắc đến là: *Ada, Algol, APL, Assembly*, *Basic*, *C*, *C++*, *C#*, *Cobol*, *Delphi*, *Fortran*, *Java*, *JavaScript, Lisp, Logo, Pascal, Perl, PHP, Prolog, Python*, *Ruby,*... Sự phát triển của ngôn ngữ lập trình gắn liền với sự phát triển của tin học. Mỗi loại ngôn ngữ phù hợp hơn với một số lớp bài toán nhất định. Cùng với tên các ngôn ngữ lập trình, các thuật ngữ thường được nhắc tới là "lập trình cấu trúc", "lập trình hướng đối tượng", "lập trình web",...

Những ngôn ngữ lập trình hiện nay thường cung cấp các thư viện bao gồm nhiều hàm hỗ trợ giao diện người dùng và các thiết bị đầu cuối. Cập nhật dữ liệu theo thời gian thực là một hướng phát triển nhằm đáp ứng các nhu cầu đồng bộ hoá nhanh dữ liệu dùng chung cho nhiều nơi hay là để thoả mãn nhu cầu cần đồng bộ hoá dữ liệu của các dịch vụ (như trong ngân hàng, hàng không và quân sự). Ngoài việc hỗ trợ cho các giao diện, ngày nay hầu hết các hệ điều hành (UNIX/Linux, Netware và Windows) đều có khả năng đa nhiệm nâng cao hiệu quả của máy tính. Do đó, các ngôn ngữ thường có thêm các hàm, thủ tục hay các biến cho phép người lập trình tận dụng điều này.

Dưới đây giới thiệu một số ngôn ngữ lập trình thông dụng: Fortran, Algol, Lisp, Cobol, Basic, Pascal, C, C++, Java,...

- **Fortran** là một ngôn ngữ lập trình được phát triển từ những năm 1950 và vẫn được dùng nhiều trong tính toán khoa học cho đến hơn nửa thế kỉ sau. Tên gọi này xuất phát từ việc ghép các từ tiếng Anh ***For****mula* ***Tran****slator* nghĩa là *dịch công thức*. Các phiên bản đầu có tên chính thức là FORTRAN. Điểm yếu của FORTRAN là thiếu hỗ trợ trực tiếp cho các kết cấu có cấu trúc, kiểu dữ liệu còn nghèo, không thuận lợi cho xử lí xâu. Fortran được phát triển ban đầu như là một ngôn ngữ thủ tục. Tuy nhiên, các phiên bản mới của Fortran đã có các tính năng hỗ trợ lập trình hướng đối tượng.
- **ALGOL** do Uỷ ban các nhà tin học châu Âu và Hoa Kì tạo ra năm 1958, là ngôn ngữ tiên phong đưa ra tập các thủ tục, định kiểu dữ liệu cực kì phong phú,... và có ảnh hưởng mạnh tới các ngôn ngữ ra đời sau.

- **LISP** do John McCarthy của Học viện Công nghệ Massachusetts tạo ra năm 1958, là ngôn ngữ đặc biệt thích hợp cho thao tác kí hiệu và xử lí danh sách thường gặp trong các bài toán tổ hợp, thích hợp cho việc chứng minh định lí. Gần đây LISP được dùng để phát triển hệ chuyên gia và các hệ thống dựa trên tri thức.
- **COBOL** ra đời năm 1959, được chấp nhận dùng cho các ứng dụng xử lí dữ liệu thương mại, kinh doanh.
- **BASIC** là ngôn ngữ được phát triển năm 1963 bởi John Kemeny và Thomas Kurtz. BASIC là ngôn ngữ còn nhiều hạn chế như thực hiện câu lệnh chủ yếu là tuần tự từ trên xuống, điều khiển chương trình chỉ nhờ lệnh IF...THEN và GOSUB.
- **PASCAL** do Niklaus Wirth phát triển dựa trên Algol năm 1970. Pascal là tên nhà toán học và triết học người Pháp Blaise Pascal. Pascal là ngôn ngữ đặc biệt thích hợp cho kiểu lập trình cấu trúc. Cho đến nay, Pascal vẫn được dùng để giảng dạy về lập trình trong nhiều trường trung học và đại học trên thế giới. Đó là ngôn ngữ cho phép mô tả thuật toán thuận tiện. Pascal cũng phục vụ nhiều ứng dụng kĩ nghệ khoa học và lập trình hệ thống. Phần lớn hệ điều hành Macintosh được viết bằng Pascal. Hệ sắp chữ TeX được Donald Knuth viết bằng ngôn ngữ mang nhiều yếu tố của Pascal. Trình biên dịch Free Pascal được viết bằng Pascal là một trình biên dịch mạnh có khả năng biên dịch cả ứng dụng cũ và mới (phân phối miễn phí dưới giấy phép GNU), hỗ trợ nhiều hệ điều hành.
- **C** là ngôn ngữ được xây dựng bởi Dennis Ritchie năm 1972 và được dùng trong hệ điều hành UNIX. Từ đó C còn được dùng trong nhiều hệ điều hành khác và trở thành một trong những ngôn ngữ phổ dụng nhất. C rất hiệu quả và được ưa chuộng để viết các phần mềm hệ thống, mặc dù nó cũng được dùng cho việc viết các ứng dụng. Ngoài ra, C cũng thường được dùng làm ngôn ngữ giảng dạy lập trình. Ngày nay, C được phát triển và mang nhiều tính năng mới làm cho nó mềm dẻo thêm.

Dennis Ritchie

- **C++** là ngôn ngữ lập trình hỗ trợ lập trình cấu trúc (thủ tục, dữ liệu trừu tượng), lập trình hướng đối tượng. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ phổ biến nhất. C++ góp phần xây dựng những ứng dụng lớn nhất hiện nay như hệ điều hành Windows, trình duyệt và máy tìm kiếm Google,… Năm 1983, Bjarne Stroustrup ở phòng thí nghiệm Bell đã phát triển C++. Trong suốt thập niên 1980 "C với các lớp" được coi là một bản nâng cao của ngôn ngữ C. Tên C++ được dùng lần đầu tiên vào tháng 12/1983. Cái tên C++ cho biết C++ là ngôn ngữ được phát triển trên cơ sở ngôn ngữ C.

- **Java** được khởi đầu bởi James Gosling và các đồng nghiệp ở Sun Microsystems năm 1991 là một phần của *Dự án Xanh*. Ban đầu ngôn ngữ này được gọi là Oak (có nghĩa là cây sồi, do bên ngoài cơ quan của ông Gosling có trồng nhiều loại cây này). Họ dự định phát triển ngôn ngữ này thay cho C++. Công ti Sun Microsystems đang giữ bản quyền và phát triển Java thường xuyên. Java được phát hành vào năm 1994, rồi nó trở nên nổi tiếng khi Netscape tuyên bố tại hội thảo *SunWorld năm 1995* là trình duyệt Navigator của họ sẽ hỗ trợ Java. Java có thể tương thích với nhiều họ máy như PC, Macintosh, tương thích với nhiều hệ điều hành như Windows, Linux. Người ta nói Java là ngôn ngữ lập trình một lần (trên một máy) nhưng có thể chạy nhiều lần (trên nhiều máy). Java được sử dụng chủ yếu để lập trình trên môi trường mạng và Internet.

James Gosling

§2. CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH

1. Các thành phần cơ bản

Mỗi ngôn ngữ lập trình thường có ba thành phần cơ bản là *bảng chữ cái*, *cú pháp* và *ngữ nghĩa*.

a) Bảng chữ cái là tập các kí tự được dùng để viết chương trình. Không được phép dùng bất kì kí tự nào ngoài các kí tự quy định trong bảng chữ cái.

Trong Pascal, bảng chữ cái bao gồm các kí tự sau:

- Các chữ cái thường và các chữ cái in hoa của bảng chữ cái tiếng Anh:

  a b c d e f g h i j k l m n o p q r s t u v w x y z
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- 10 chữ số thập phân Ả Rập: 0 1 2 3 4 5 6 7 8 9

- Các kí tự đặc biệt:

+	–	*	/	=		<	>	[	]	.	,
;	#	^	$	@	&	(	)	{	}	:	'
dấu cách (mã ASCII 32)										_	

Bảng chữ cái của các ngôn ngữ lập trình nói chung không khác nhau nhiều. Ví dụ, bảng chữ cái của ngôn ngữ lập trình C++ chỉ khác Pascal là có sử dụng thêm các kí tự như dấu nháy kép ("), dấu sổ ngược (\), dấu chấm than (!).

b) Cú pháp là bộ quy tắc để viết chương trình. Dựa vào chúng, người lập trình và chương trình dịch biết được tổ hợp nào của các kí tự trong bảng chữ cái là hợp lệ và tổ hợp nào là không hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy thực hiện.

c) Ngữ nghĩa xác định ý nghĩa thao tác cần phải thực hiện, ứng với tổ hợp kí tự dựa vào ngữ cảnh của nó.

Ví dụ

Phần lớn các ngôn ngữ lập trình đều sử dụng dấu cộng (+) để chỉ phép cộng. Xét các biểu thức:

```
A + B                    (1)
I + J                    (2)
```

Giả thiết *A*, *B* là các đại lượng nhận giá trị thực và *I*, *J* là các đại lượng nhận giá trị nguyên. Khi đó dấu "+" trong biểu thức (1) được hiểu là cộng hai số thực, dấu "+" trong biểu thức (2) được hiểu là cộng hai số nguyên. Như vậy, ngữ nghĩa dấu "+" trong hai ngữ cảnh khác nhau là khác nhau.

Tóm lại, cú pháp cho biết cách viết một chương trình hợp lệ, còn ngữ nghĩa xác định ý nghĩa của các tổ hợp kí tự trong chương trình.

Các lỗi cú pháp được chương trình dịch phát hiện và thông báo cho người lập trình biết. Chỉ có các chương trình không còn lỗi cú pháp mới dịch được sang ngôn ngữ máy.

Các lỗi ngữ nghĩa khó phát hiện hơn. Phần lớn các lỗi ngữ nghĩa chỉ được phát hiện khi thực hiện chương trình trên dữ liệu cụ thể.

2. Một số khái niệm

a) Tên

Mọi đối tượng trong chương trình đều phải được đặt tên theo quy tắc của ngôn ngữ lập trình và từng chương trình dịch cụ thể.

Trong Turbo Pascal, tên là một dãy liên tiếp *không quá 127 kí tự bao gồm chữ số, chữ cái hoặc dấu gạch dưới và bắt đầu bằng chữ cái hoặc dấu gạch dưới*. Trong chương trình dịch Free Pascal, tên có thể có độ dài tới 255 kí tự.

Ví dụ, trong ngôn ngữ Pascal:

- Các tên đúng: `A`
  `R21`
  `P21_c`
  `_45`
- Các tên sai:
  `A BC` (chứa dấu cách)
  `6Pq` (bắt đầu bằng chữ số)
  `X#Y` (chứa kí tự "#" không hợp lệ)

Ngôn ngữ Pascal không phân biệt chữ hoa, chữ thường trong tên. Một số ngôn ngữ lập trình khác (ví dụ như C++) phân biệt chữ hoa, chữ thường. Ví dụ, *AB* và *Ab* là một tên trong Pascal, nhưng lại là hai tên khác nhau trong C++.

Nhiều ngôn ngữ lập trình, trong đó có Pascal, phân biệt ba loại tên:

- Tên dành riêng;
- Tên chuẩn;
- Tên do người lập trình đặt.

Tên dành riêng

Một số tên được ngôn ngữ lập trình quy định dùng với *ý nghĩa riêng xác định*, người lập trình không được sử dụng với ý nghĩa khác. Những tên này được gọi là *tên dành riêng* (còn được gọi là *từ khoá*).

Ví dụ. Một số tên dành riêng:

Trong Pascal: `program`, `uses`, `const`, `type`, `var`, `begin`, `end`.

Trong C++: `main`, `include`, `if`, `while`, `void`.

Tên chuẩn

Một số tên được ngôn ngữ lập trình dùng với *ý nghĩa nhất định nào đó*. Những tên này được gọi là *tên chuẩn*. Tuy nhiên, người lập trình có thể khai báo và dùng chúng với ý nghĩa và mục đích khác.

Ý nghĩa của các tên chuẩn được quy định trong các *thư viện* của ngôn ngữ lập trình.

Ví dụ. Một số tên chuẩn

– Trong Pascal:

`abs`	`integer`	`real`
`sqr`	`longint`	`extended`
`sqrt`	`byte`	`break`

– Trong C++:

`cin`	`cout`	`getchar`

Tên do người lập trình đặt

Tên do người lập trình đặt được dùng với ý nghĩa riêng, xác định bằng cách khai báo trước khi sử dụng. Các tên này không được trùng với tên dành riêng.

Ví dụ

Tên do người lập trình đặt:

```
A1
DELTA
CT_Vidu
```

b) Hằng và biến

Hằng

Hằng là đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.

Trong các ngôn ngữ lập trình thường có các hằng số học, hằng lôgic, hằng xâu.

- Hằng số học là các số nguyên hay số thực (dấu phẩy tĩnh hoặc dấu phẩy động).
- Hằng lôgic là giá trị *đúng* hoặc *sai* tương ứng với *true* hoặc *false*.
- Hằng xâu là dãy kí tự trong bộ mã ASCII. Khi viết, dãy kí tự này được đặt trong cặp dấu nháy (Pascal dùng dấu nháy đơn, còn C++ dùng dấu nháy kép).

Ví dụ

– Hằng số học: `2` `0` `-5` `+18`
`1.5` `-22.36` `+3.14159` `0.5`
`-2.236E01` `1.0E-6`

– Hằng lôgic:

+ Trong Pascal: `TRUE` `FALSE`

– Hằng xâu:

+ Trong Pascal: `'Information'` `'Lop 11A'`
+ Trong C++: `"Information"` `"Lop 11A"`

Chú ý: Hằng dấu nháy đơn trong Pascal được viết là ''''. Để có xâu tiếng Anh *I'm a student*, trong Pascal cần viết là 'I''m a student'.

Biến

Biến là đại lượng được đặt tên, dùng để lưu trữ giá trị và giá trị có thể được thay đổi trong quá trình thực hiện chương trình.

Tuỳ theo cách lưu trữ và xử lí, Pascal phân biệt nhiều loại biến. Các biến dùng trong chương trình đều phải khai báo. Việc khai báo biến sẽ được trình bày ở các phần sau.

c) *Chú thích*

Có thể đặt các đoạn chú thích trong chương trình nguồn. Các chú thích này giúp cho người đọc chương trình nhận biết ý nghĩa của chương trình đó dễ hơn. Chú thích không ảnh hưởng đến nội dung chương trình nguồn và được chương trình dịch bỏ qua.

Trong Pascal các đoạn chú thích được đặt giữa cặp dấu { và } hoặc (* và *). Một trong những cách tạo chú thích trong C++ là đặt chúng giữa cặp dấu /* và */.

TÓM TẮT

- Cần có chương trình dịch để chuyển chương trình nguồn thành chương trình đích.
- Có hai loại chương trình dịch: thông dịch và biên dịch.
- Các thành phần của ngôn ngữ lập trình: bảng chữ cái, cú pháp và ngữ nghĩa.
- Mọi đối tượng trong chương trình đều phải được đặt tên:
  - Tên dành riêng: Được dùng với ý nghĩa riêng, không được dùng với ý nghĩa khác.
  - Tên chuẩn: Tên dùng với ý nghĩa nhất định, khi cần dùng với ý nghĩa khác thì phải khai báo.
  - Tên do người lập trình đặt: Cần khai báo trước khi sử dụng.
- Hằng: Đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.
- Biến: Đại lượng được đặt tên. Giá trị của biến có thể thay đổi trong quá trình thực hiện chương trình.

CÂU HỎI VÀ BÀI TẬP

1. Tại sao người ta phải xây dựng các ngôn ngữ lập trình bậc cao?
2. Chương trình dịch là gì? Tại sao cần phải có chương trình dịch?
3. Biên dịch và thông dịch khác nhau như thế nào?
4. Hãy cho biết các điểm khác nhau giữa tên dành riêng và tên chuẩn.
5. Hãy tự viết ra ba tên đúng theo quy tắc của Pascal.
6. Hãy cho biết những biểu diễn nào dưới đây không phải là biểu diễn hằng trong Pascal và chỉ rõ lỗi trong từng trường hợp:

a) 150.0	b) –22	c) 6,23	d) '43'
e) A20	f) 1.06E–15	g) 4+6	h) 'C
i) 'TRUE'			

Bài đọc thêm 2

NGÔN NGỮ PASCAL

1. Vài nét về tác giả của ngôn ngữ Pascal

Giáo sư Niklaus Wirth, tác giả của ngôn ngữ Pascal sinh năm 1934 tại Thuỵ Sĩ. Ông tốt nghiệp Đại học Công nghệ Liên bang Thuỵ Sĩ (ETH) tại thành phố quê hương Zurich vào năm 1959. Ông nhận bằng Thạc sĩ tại trường Đại học Tổng hợp Laval ở Quebec, Canada năm 1960.

Năm 1963, tại Đại học Tổng hợp California (Mĩ) dưới sự lãnh đạo của giáo sư Harry Huskey ông thực hiện đề án mở rộng ngôn ngữ Algol-60 (ngôn ngữ Euler) và bảo vệ luận án tiến sĩ.

Trong các năm 1963-1967, ông giảng dạy tại Đại học Tổng hợp Stanford (Mĩ). Cũng trong thời gian này ông được mời vào nhóm chuyên gia quốc tế IFIP thiết kế Algol-68.

Năm 1967, Wirth trở về nước và giảng dạy tại Tổng hợp Zurich.

Năm 1968, ông chuyển sang ETH, tại đây ông bắt đầu tham gia thiết kế Pascal.

Năm 1970, chương trình dịch Pascal đầu tiên được hoàn thành.

Trong thời gian 1978-1981, Wirth lãnh đạo dự án thiết kế ngôn ngữ Modula-2, máy tính cá nhân 16-bit Lilith dựa trên nó và hệ điều hành OS Medos. Tất cả các chương trình, kể cả chương trình hệ thống, được thực hiện hoàn toàn trên Modula-2.

Năm 1984, do công lao to lớn trong việc phát triển các ngôn ngữ lập trình và thiết kế máy tính cá nhân Lilith, ông được giải thưởng Alan Turing – giải thưởng cao quý nhất trong giới Tin học, mà về ý nghĩa được coi là tương đương với giải Nobel.

Trong thời gian 1986-1989, Wirth lãnh đạo dự án phát triển ngôn ngữ Oberon, hệ điều hành hướng đối tượng Oberon và trạm làm việc 32-bit Ceres. Rất nhiều ý tưởng của dự án này được các đồng nghiệp từ phòng thí nghiệm Sun Labs sử dụng cho ngôn ngữ và công nghệ Java.

Từ năm 1990 ông lãnh đạo Viện Các hệ thống máy tính tại ETH.

Năm 1999, ông nghỉ hưu và trở thành Giáo sư danh dự của ETH.

2. Pascal – Ngôn ngữ của học đường?

Pascal! Hỏi có người lập trình nào không biết ngôn ngữ này? Thành công vang dội của nó bắt đầu vào những năm 1980, thời kì của cuộc cách mạng trong công nghiệp máy tính và giai đoạn nở rộ của lập trình có cấu trúc. Có thể nói Pascal xứng đáng là điểm khởi đầu cho một kỉ nguyên mới của các ngôn ngữ lập trình.

Pascal – Sự ra đời và đặc điểm

Vào đầu năm 1971, bản mô tả ngôn ngữ mới của Viện Công nghệ Liên bang Thuỵ Sĩ được công bố trong số đầu tiên của tạp chí Acta Informatica. Sự ra đời của Pascal có thể được tính từ thời điểm này.

Tác giả của nó, Giáo sư Niklaus Wirth trở nên nổi tiếng vì sự xuất hiện của Pascal. Những dự án sau này của ông chứng minh hùng hồn cho thế giới rằng chìa khoá tới các bí mật của máy tính chính là ở sự kết hợp hài hoà giữa Toán học, Công nghệ và Lập trình. Và nếu tiếp cận vấn đề một cách hợp lí thì có thể tạo ra các ngôn ngữ, hệ điều hành và ngay cả các máy tính tuyệt vời vượt các chuẩn công nghiệp,... chỉ bằng sức lực của những sinh viên.

Người ta thường nói đến các điểm khác biệt của Pascal so với những ngôn ngữ khác, như ngôn ngữ C. Nhưng chính Dennis Ritchie, tác giả của C đã phát biểu (1993): "Tôi khẳng định rằng Pascal rất gần với C. Hai ngôn ngữ này khác biệt về chi tiết, nhưng về cơ sở chúng là giống nhau... Khi nhìn vào các kiểu dữ liệu, cũng như các phép toán trên chúng, ta có thể phát hiện ra những sự giống nhau rất lớn, mặc dù rằng ý đồ của Wirth khi tạo ra Pascal rất khác với ý đồ của chúng tôi khi tạo ra C. Wirth tạo ngôn ngữ để giảng dạy và do vậy tất nhiên cần đáp ứng các yêu cầu sư phạm".

Khác với C, Pascal không được tạo ra để làm ngôn ngữ lập trình hệ thống. Để đề cao tính đơn giản và hiệu quả dựa trên mức độ hiểu về khoa học lập trình thời bấy giờ, Wirth chủ tâm chấp nhận các hạn chế của ngôn ngữ, trước tiên trong các vấn đề liên quan đến thế giới bên ngoài (vào/ra và các công cụ phụ thuộc hệ thống). Mặc dù vậy, nếu nghĩ rằng Pascal là ngôn ngữ chỉ dành để giảng dạy thì sẽ sai lầm. Hãy nghe chính ý kiến của Wirth về vấn đề này (1984): "Có người cho rằng Pascal được thiết kế như một ngôn ngữ để giảng dạy. Mặc dù điều này là đúng, nhưng việc sử dụng nó để giảng dạy không phải là mục đích duy nhất. Thực tế, tôi không tin vào sự thành công của việc áp dụng trong khi học các công cụ và phương pháp mà không thể sử dụng để giải quyết các bài toán thực tế. Theo các tiêu chuẩn ngày nay thì Pascal có những nhược điểm rõ ràng khi lập trình các hệ thống lớn, nhưng 15 năm trước, nó là thoả hiệp hợp lí giữa cái mong muốn và hiệu quả".

Pascal và lập trình có cấu trúc

Ngôn ngữ Pascal được Wirth tạo ra dưới ảnh hưởng các tư tưởng của C. A. R. Hoare, đăng trong công trình "Bàn về cấu trúc dữ liệu" (Notes on Data Structuring, Academic Press, 1972). Đóng góp của nhà bác học người Anh lớn tới mức có thể gọi ông là cha đỡ đầu của Pascal.

Pascal được coi như khởi đầu của kỉ nguyên lập trình cấu trúc. Tất cả bắt đầu từ bài báo của chuyên gia người Hà Lan E. W. Dijkstra "Lập trình cấu trúc" (Structured Programming, 1969). Trong bài báo này ông đề xuất hạn chế các cấu trúc điều khiển chương trình chỉ ở ba dạng là tuần tự, rẽ nhánh và lặp. Từ đó suy ra rằng câu lệnh chuyển vô điều kiện (goto) trong các ngôn ngữ ALGOL và PL/1 (rất phổ biến thời bấy giờ) là hoàn toàn không cần thiết. Thật sự là Wirth cũng không dám loại câu lệnh này khỏi Pascal. Nhưng điều chủ yếu nằm ở chỗ khác: Lập trình cấu trúc liên quan đến nguyên tắc "từ trên xuống dưới" (làm mịn từng bước), yêu cầu tính cấu trúc của điều khiển và dữ liệu, dựa vào sự đơn giản và cơ sở toán học mà tăng độ tin cậy của phần mềm. Tất cả những điều này đều khả thi nhờ các khả năng của Pascal.

Về ý đồ khi xây dựng Pascal, Wirth viết: "Điểm mới của Pascal là đưa ra các cấu trúc và kiểu dữ liệu phong phú, cũng giống như ALGOL đưa ra các loại cấu trúc điều khiển. Trong ALGOL chỉ có ba kiểu dữ liệu cơ sở: các số nguyên và thực, giá trị chân lí, mảng; Pascal đã đưa thêm các kiểu dữ liệu cơ sở và còn cho khả năng xác định những kiểu cơ sở mới (kiểu liệt kê, kiểu miền con), cũng như các dạng cấu trúc dữ liệu mới: bản ghi, tập hợp, tệp, mà một số trong chúng đã có trong COBOL. Và tất nhiên, quan trọng nhất là tính đệ quy trong việc mô tả các cấu trúc và hệ quả của điều này là khả năng kết hợp và lồng các cấu trúc".

Pascal có tiếp tục tồn tại?

Để kết luận, xin trích dẫn lời của Dennis Ritchie tác giả ngôn ngữ C: "Pascal là một ngôn ngữ thanh lịch. Nó vẫn tiếp tục tồn tại. Nó đã khởi nguồn cho không ít ngôn ngữ đàn em và có ảnh hưởng sâu sắc đến việc thiết kế các ngôn ngữ lập trình nói chung".

Chương 11

CHƯƠNG TRÌNH ĐƠN GIẢN

§3. CẤU TRÚC CHƯƠNG TRÌNH

1. Cấu trúc chung

Nói chung, chương trình được viết bằng một ngôn ngữ lập trình bậc cao thường gồm *phần khai báo* và *phần thân*. Phần thân chương trình nhất thiết phải có. Phần khai báo có thể có hoặc không tuỳ theo từng chương trình cụ thể.

Khi diễn giải cú pháp của ngôn ngữ lập trình người ta thường sử dụng ngôn ngữ tự nhiên. Các diễn giải bằng ngôn ngữ tự nhiên được đặt giữa cặp dấu < và >. Các thành phần của chương trình có thể có hoặc không được đặt trong cặp dấu [và].

Với quy ước trên, cấu trúc của một chương trình có thể được mô tả như sau:

[< *phần khai báo* >]

<*phần thân*>

2. Các thành phần của chương trình

a) Phần khai báo

Có thể có các khai báo cho: tên chương trình, thư viện, hằng, biến và chương trình con.

Khai báo tên chương trình

Phần này có thể có hoặc không. Với Pascal, nếu có, phần khai báo tên chương trình bắt đầu bằng từ khoá **program**, tiếp đến là tên chương trình:

program <*tên chương trình*>;

trong đó, *tên chương trình* là tên do người lập trình đặt theo đúng quy định về tên.

Ví dụ

```
program Phuong_trinh_B2;
program Vi_du;
```

Khai báo thư viện

Mỗi ngôn ngữ lập trình thường có sẵn một số thư viện cung cấp một số chương trình thông dụng đã được lập sẵn. Để sử dụng các chương trình đó cần khai báo thư viện chứa nó.

Ví dụ. Khai báo thư viện

– Trong Pascal:

```
uses crt;
```

– Trong C++:

```
#include <stdio.h>
#include <conio.h >
```

Thư viện *crt* trong Pascal hoặc *stdio.h* và *conio.h* trong C++ cung cấp các chương trình có sẵn để làm việc với màn hình và bàn phím. Ví dụ, muốn xoá những gì đang có trên màn hình:

– Trong Pascal, sau khi khai báo thư viện *crt*, ta dùng lệnh:

```
clrscr;
```

– Trong C++, sau khi khai báo thư viện *conio.h*, ta dùng lệnh:

```
clrscr();
```

Khai báo hằng

Ví dụ. Khai báo hằng

– Trong Pascal:

```
const MaxN = 1000;
      PI = 3.1416;
      KQ = 'Ket qua:';
```

– Trong C++:

```
const int MaxN = 1000;
const float PI = 3.1416;
const char* KQ = "ketqua:";
```

Khai báo hằng thường được sử dụng cho những giá trị xuất hiện nhiều lần trong chương trình.

Khai báo biến

Tất cả các biến dùng trong chương trình đều phải được đặt tên và khai báo cho chương trình dịch biết để lưu trữ và xử lí. Biến chỉ nhận một giá trị tại mỗi thời điểm thực hiện chương trình được gọi là *biến đơn*.

Ví dụ

Khi khảo sát phương trình đường thẳng $ax + by + c = 0$, các hệ số a, b, c có thể được khai báo như những biến đơn.

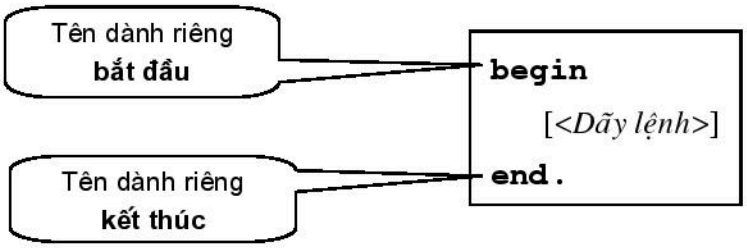
Cách khai báo biến được trình bày riêng trong §5.

Khai báo và sử dụng chương trình con được trình bày trong chương VI.

b) Phần thân chương trình

Dãy lệnh trong phạm vi được xác định bởi cặp dấu hiệu mở đầu và kết thúc tạo thành thân chương trình.

Ví dụ. Thân chương trình trong Pascal:



3. Ví dụ chương trình đơn giản

Dưới đây xét một vài ví dụ về những chương trình đơn giản.

Ví dụ 1. Chương trình sau thực hiện việc đưa ra màn hình thông báo "*Xin chao cac ban!*".

Trong Pascal	Trong C++

`program vi_du;`
`begin`
`  writeln('Xin chao cac ban!');`
`end.`
	`#include <stdio.h>`
	`void main()`
	`{`
	`  printf("Xin chao cac ban!");`
	`}`

– Phần khai báo chỉ có khai báo tên chương trình gồm tên dành riêng *program* và tên chương trình là *vi_du*. – Phần thân chương trình chỉ có một câu lệnh *writeln*, đưa thông báo ra màn hình.	– Phần khai báo chỉ có một câu lệnh *include* khai báo thư viện *stdio.h*. – Phần thân chương trình chỉ có một câu lệnh *printf* đưa thông báo ra màn hình.

Ví dụ 2. Chương trình Pascal sau đưa các thông báo "*Xin chao cac ban!*" và "*Moi cac ban lam quen voi Pascal*" ra màn hình.

```
begin
    writeln('Xin chao cac ban!');
    writeln('Moi cac ban lam quen voi Pascal');
end.
```

Chương trình trên không có phần khai báo. Phần thân chương trình có hai câu lệnh đưa hai thông báo tương ứng ra màn hình.

§4. MỘT SỐ KIỂU DỮ LIỆU CHUẨN

Các bài toán trong thực tế thường có dữ liệu vào và kết quả ra thuộc những kiểu dữ liệu quen biết như số nguyên, số thực, kí tự,... Khi cần lập trình cho những bài toán như vậy, người lập trình sử dụng các kiểu dữ liệu đó thường gặp một số hạn chế nhất định, phụ thuộc vào các yếu tố như dung lượng bộ nhớ, khả năng xử lí của CPU,...

Vì vậy, mỗi ngôn ngữ lập trình thường cung cấp một số kiểu dữ liệu chuẩn cho biết phạm vi giá trị có thể lưu trữ, dung lượng bộ nhớ cần thiết để lưu trữ và các phép toán tác động lên dữ liệu. Dưới đây xét một số kiểu dữ liệu chuẩn thường dùng cho các biến đơn trong Pascal.

1. Kiểu nguyên

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
`byte`	1 byte	từ 0 đến 255
`integer`	2 byte	từ –32768 đến 32767
`word`	2 byte	từ 0 đến 65535
`longint`	4 byte	từ –2147483648 đến 2147483647

2. Kiểu thực

Có nhiều kiểu dùng để khai báo các đại lượng nhận giá trị là số thực. Thường dùng hơn cả là các kiểu được liệt kê trong bảng sau:

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
`real`	6 byte	0 hoặc có giá trị tuyệt đối nằm trong phạm vi từ $2,9 \times 10^{-39}$ đến $1,7 \times 10^{38}$
`extended`	10 byte	0 hoặc có giá trị tuyệt đối nằm trong phạm vi từ 10^{-4932} đến 10^{4932}

3. Kiểu kí tự

Ta hiểu kí tự là các kí tự thuộc bộ mã ASCII gồm 256 kí tự có mã ASCII thập phân từ 0 đến 255.

Ví dụ, kí tự *A* có mã ASCII là 65, kí tự *a* có mã ASCII là 97. Kí tự đặc biệt, dùng để thể hiện sự ngăn cách giữa hai từ viết liên tiếp trong các văn bản, là dấu cách. Dấu cách được gõ bằng phím **Space** – phím dài nhất trên bàn phím và có mã ASCII bằng 32.

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
`char`	1 byte	256 kí tự trong bộ mã ASCII

4. Kiểu lôgic

Kiểu	Bộ nhớ lưu trữ một giá trị	Phạm vi giá trị
`boolean`	1 byte	true hoặc false

Ghi chú: Người lập trình cần tìm hiểu đặc trưng của các kiểu dữ liệu chuẩn được xác định bởi bộ dịch và sử dụng để khai báo biến.

§5. KHAI BÁO BIẾN

Như đã nêu ở trên, mọi biến dùng trong chương trình đều cần khai báo tên và kiểu dữ liệu. Tên biến dùng để xác lập quan hệ giữa biến với địa chỉ bộ nhớ nơi lưu trữ giá trị biến. Mỗi biến chỉ được khai báo một lần. Trong phần này ta chỉ xét khai báo các biến đơn.

Trong Pascal, khai báo biến bắt đầu bằng từ khoá *var* có dạng:

var <*danh sách biến*>: <*kiểu dữ liệu*>;

trong đó:

- *danh sách biến* là một hoặc nhiều tên biến, các tên biến được viết cách nhau bởi dấu phẩy;
- *kiểu dữ liệu* thường là một trong các kiểu dữ liệu chuẩn hoặc kiểu dữ liệu do người lập trình định nghĩa (được trình bày trong chương IV).

Sau từ khoá *var* có thể khai báo nhiều danh sách biến khác nhau, tức là cấu trúc:

<danh sách biến>: <kiểu dữ liệu>;

có thể xuất hiện nhiều lần.

Ví dụ 1

Giả sử trong chương trình cần các biến thực *A*, *B*, *C*, *D*, *X*1, *X*2 và các biến nguyên *M*, *N*. Khi đó có thể khai báo các biến đó như sau:

```
var
    A, B, C, D, X1, X2: real;
    M, N: integer;
```

Ví dụ 2

Xét khai báo biến:

```
var
    X, Y, Z: real;
    C: char;
    I, J: byte;
    N: word;
```

Trong khai báo này có ba biến thực *X*, *Y*, *Z*. Bộ nhớ cấp phát cho ba biến này là 18 byte ($3 \times 6 = 18$). *C* là biến kí tự và bộ nhớ dành cho nó là 1 byte. Các biến *I*, *J* nhận giá trị nguyên trong phạm vi từ 0 đến 255 và bộ nhớ dành cho mỗi biến là 1 byte. Biến *N* cũng nhận các giá trị nguyên, nhưng trong phạm vi từ 0 đến 65535. Bộ nhớ cấp phát cho biến *N* là 2 byte. Như vậy, tổng bộ nhớ dành cho các biến đã khai báo là:

$$18 + 1 + 2 + 2 = 23 \text{ (byte)}.$$

Một số chú ý khi khai báo biến:

- Cần đặt tên biến sao cho gợi nhớ đến ý nghĩa của biến đó. Điều này rất có lợi cho việc đọc, hiểu và sửa đổi chương trình khi cần thiết.

  Ví dụ, cần đặt tên hai biến biểu diễn điểm toán, điểm tin thì không nên vì ngắn gọn mà đặt tên biến là *d*1, *d*2 mà nên đặt là *dtoan*, *dtin*.
- Không nên đặt tên biến quá ngắn hay quá dài, dễ mắc lỗi khi viết nhiều lần tên biến. Ví dụ, không nên dùng *d*1, *d*2 hay *diemmontoan*, *diemmontin* cho điểm toán, điểm tin của học sinh.
- Khi khai báo biến cần đặc biệt lưu ý đến phạm vi giá trị của nó. Ví dụ, khi khai báo biến biểu diễn số học sinh của một lớp có thể sử dụng kiểu byte, nhưng biến biểu diễn số học sinh của toàn trường thì phải thuộc kiểu word.

§6. PHÉP TOÁN, BIỂU THỨC, CÂU LỆNH GÁN

Để mô tả các thao tác trong thuật toán, mỗi ngôn ngữ lập trình đều xác định và sử dụng một số khái niệm cơ bản: phép toán, biểu thức, gán giá trị cho biến.

Dưới đây sẽ xét các khái niệm đó trong Pascal.

1. Phép toán

Tương tự trong toán học, trong các ngôn ngữ lập trình đều có những phép toán số học như cộng, trừ, nhân, chia trên các đại lượng thực, các phép toán chia nguyên và lấy phần dư, các phép toán quan hệ,…

Bảng dưới đây là kí hiệu các phép toán đó trong toán và trong Pascal:

Phép toán	Trong toán học	Trong Pascal
Các phép toán số học với số nguyên	+ (cộng), – (trừ), × (nhân), div (chia nguyên), mod (lấy phần dư)	+, –, *, div, mod
Các phép toán số học với số thực	+ (cộng), – (trừ), × (nhân), : (chia)	+, –, *, /
Các phép toán quan hệ	< (nhỏ hơn), ≤ (nhỏ hơn hoặc bằng), > (lớn hơn), ≥ (lớn hơn hoặc bằng), = (bằng), ≠ (khác)	<, <=, >, >=, =, <>
Các phép toán lôgic	¬ (phủ định), ∨ (hoặc), ∧ (và)	not, or, and

Chú ý: – Kết quả của các phép toán quan hệ cho giá trị lôgic.

– Một trong những ứng dụng của phép toán lôgic là để tạo ra các biểu thức phức tạp từ các quan hệ đơn giản.

2. Biểu thức số học

Trong lập trình, biểu thức số học là một biến kiểu số hoặc một hằng số hoặc các biến kiểu số và các hằng số liên kết với nhau bởi một số hữu hạn phép toán số học, các dấu ngoặc tròn (và) tạo thành một biểu thức có dạng tương tự như cách viết trong toán học với những quy tắc sau:

- Chỉ dùng cặp ngoặc tròn để xác định trình tự thực hiện phép toán trong trường hợp cần thiết;
- Viết lần lượt từ trái qua phải;
- Không được bỏ qua dấu nhân (*) trong tích.

Các phép toán được thực hiện theo thứ tự:

- Thực hiện các phép toán trong ngoặc trước;
- Trong dãy các phép toán không chứa ngoặc thì thực hiện từ trái sang phải, theo thứ tự các phép toán nhân (*), chia (/), chia nguyên (div), lấy phần dư (mod) thực hiện trước và các phép toán cộng (+), trừ (–) thực hiện sau.

Ví dụ

Biểu thức trong toán học	Biểu thức trong Pascal
5a+6b	5*a + 6*b
$\frac{xy}{z}$	x*y/z
$Ax^2 + Bx + C$	A*x*x + B*x + C
$\frac{x+y}{x-\frac{1}{2}} - \frac{x-z}{xy}$	(x + y)/(x – 1/2) – (x – z)/(x*y)

Chú ý: – Nếu biểu thức chứa một hằng hay biến kiểu thực thì ta có biểu thức số học thực, giá trị của biểu thức cũng thuộc kiểu thực.

– Trong một số trường hợp nên dùng biến trung gian để có thể tránh được việc tính một biểu thức nhiều lần.

3. Hàm số học chuẩn

Để lập trình được dễ dàng, thuận tiện hơn, các ngôn ngữ lập trình đều có thư viện chứa một số chương trình tính giá trị những hàm toán học thường dùng. Các chương trình như vậy được gọi là các *hàm số học chuẩn*. Mỗi hàm chuẩn có tên chuẩn riêng. Đối số của hàm là một hay nhiều biểu thức số học và được đặt trong cặp ngoặc tròn (và) sau tên hàm. Bản thân hàm chuẩn cũng được coi là một biểu thức số học và nó có thể tham gia vào biểu thức số học như một toán hạng (giống như biến và hằng). Kết quả của hàm có thể là nguyên hoặc thực hay phụ thuộc vào kiểu của đối số.

Bảng dưới đây cho biết một số hàm chuẩn thường dùng.

Hàm	Biểu diễn toán học	Biểu diễn trong Pascal	Kiểu đối số	Kiểu kết quả
Bình phương	x^2	**sqr**(x)	Thực hoặc nguyên	Theo kiểu của đối số
Căn bậc hai	$\sqrt{x}$	**sqrt**(x)	Thực hoặc nguyên	Thực
Giá trị tuyệt đối	$\|x\|$	**abs**(x)	Thực hoặc nguyên	Theo kiểu của đối số
Lôgarit tự nhiên	lnx	**ln**(x)	Thực	Thực
Luỹ thừa của số e	e^x	**exp**(x)	Thực	Thực
Sin	sinx	**sin**(x)	Thực	Thực
Cos	cosx	**cos**(x)	Thực	Thực

Ví dụ

Biểu thức toán học $\dfrac{-b+\sqrt{b^2-4ac}}{2a}$ trong Pascal có thể viết dưới dạng:

```
(-b+sqrt(b*b - 4*a*c))/(2*a)
```

hoặc

```
(-b+sqrt(sqr(b)- 4*a*c))/2/a
```

Ngoài những hàm số học chuẩn trên, còn có các hàm chuẩn khác được giới thiệu trong những phần sau.

4. Biểu thức quan hệ

Hai biểu thức cùng kiểu liên kết với nhau bởi phép toán quan hệ cho ta một biểu thức quan hệ.

Biểu thức quan hệ có dạng:

<biểu thức 1> <phép toán quan hệ> <biểu thức 2>

trong đó, *biểu thức 1* và *biểu thức 2* cùng là xâu hoặc cùng là biểu thức số học.

Ví dụ

```
x < 5
i+1 >= 2*j
```

Biểu thức quan hệ được thực hiện theo trình tự:

- Tính giá trị các biểu thức.
- Thực hiện phép toán quan hệ.

Kết quả của biểu thức quan hệ là giá trị lôgic: *true* (đúng) hoặc *false* (sai).

Trong ví dụ trên, nếu x có giá trị 3, thì biểu thức $x < 5$ có giá trị *true*. Nếu i có giá trị 2 và j có giá trị 3 thì biểu thức $i + 1 >= 2*j$ sẽ cho giá trị *false*.

Ví dụ

Điều kiện để điểm M có toạ độ (x, y) thuộc hình tròn tâm $I(a, b)$, bán kính R là:

```
sqrt((x-a)*(x-a) + (y-b)*(y-b)) <= R
```

hoặc

```
sqr(x-a) + sqr(y-b) <= R*R
```

5. Biểu thức lôgic

Biểu thức lôgic đơn giản là biến lôgic hoặc hằng lôgic.

Biểu thức lôgic là các biểu thức lôgic đơn giản, các biểu thức quan hệ liên kết với nhau bởi phép toán lôgic. Giá trị biểu thức lôgic là *true* hoặc *false* (xem phụ lục A). Các biểu thức quan hệ thường được đặt trong cặp ngoặc (và).

Phép toán ***not*** được viết trước biểu thức cần phủ định, ví dụ:

not $(x < 1)$ thể hiện phát biểu "x không nhỏ hơn 1" và điều này tương đương với biểu thức quan hệ $x >= 1$.

Các phép toán ***and*** và ***or*** dùng để kết hợp nhiều biểu thức lôgic hoặc quan hệ thành một biểu thức, thường được dùng để diễn tả các điều kiện phức tạp.

Ví dụ 1

Để thể hiện điều kiện $5 \le x \le 11$, trong Pascal cần phải tách thành phát biểu dưới dạng "$5 \le x$ và $x \le 11$":

```
(5 <= x) and (x <= 11)
```

Ví dụ 2

Giả thiết *M* và *N* là hai biến nguyên. Điều kiện xác định *M* và *N* đồng thời chia hết cho 3 hay đồng thời không chia hết cho 3 được thể hiện trong Pascal như sau:

```
((M mod 3 = 0) and (N mod 3 = 0)) or ((M mod 3 <>0) and (N mod 3 <> 0))
```

6. Câu lệnh gán

Lệnh gán là một trong những lệnh cơ bản nhất của các ngôn ngữ lập trình.

Trong Pascal câu lệnh gán có dạng:

<tên biến> := *<biểu thức>*;

Trong trường hợp đơn giản, *tên biến* là tên của biến đơn. Kiểu của giá trị biểu thức phải phù hợp với kiểu của biến.

Chức năng của lệnh gán là đặt cho biến có tên ở vế trái dấu ":=" giá trị mới bằng giá trị của biểu thức ở vế phải.

Ví dụ

```
x1:= (-b - sqrt(b*b - 4*a*c))/(2*a);
x2:= -b/a - x1;
z:= z - 1;
i:= i + 1;
```

Trong ví dụ trên, ý nghĩa của lệnh gán thứ ba là giảm giá trị của biến *z* một đơn vị. Ý nghĩa của lệnh gán thứ tư là tăng giá trị của biến *i* lên một đơn vị.

§7. CÁC THỦ TỤC CHUẨN VÀO/RA ĐƠN GIẢN

Để khởi tạo giá trị ban đầu cho biến, ta có thể dùng lệnh gán để gán một giá trị cho biến. Như vậy, mỗi chương trình luôn làm việc với một bộ dữ liệu vào. Để chương trình có thể làm việc với nhiều bộ dữ liệu vào khác nhau, thư viện của các ngôn ngữ lập trình cung cấp một số chương trình dùng để đưa dữ liệu vào và đưa dữ liệu ra.

Những chương trình đưa dữ liệu vào cho phép đưa dữ liệu từ bàn phím hoặc từ đĩa vào gán cho các biến, làm cho chương trình trở nên linh hoạt, có thể tính toán với nhiều bộ dữ liệu đầu vào khác nhau. Kết quả tính toán được lưu trữ tạm thời trong bộ nhớ. Những chương trình đưa dữ liệu ra dùng để đưa các kết quả này ra màn hình, in ra giấy hoặc lưu trên đĩa.

Các chương trình đưa dữ liệu vào và ra đó được gọi chung là các *thủ tục chuẩn vào/ra đơn giản*.

Trong phần này, ta sẽ xét các *thủ tục chuẩn vào/ra đơn giản* của Pascal để nhập dữ liệu vào từ bàn phím và đưa thông tin ra màn hình.

1. Nhập dữ liệu vào từ bàn phím

Việc nhập dữ liệu từ bàn phím được thực hiện bằng thủ tục chuẩn:

```
read(<danh sách biến vào>);
```

hoặc

```
readln(<danh sách biến vào>);
```

trong đó *danh sách biến vào* là *một* hoặc *nhiều* tên biến đơn (trừ biến kiểu boolean). Trong trường hợp nhiều biến thì các tên biến được viết cách nhau bởi *dấu phẩy*.

Ví dụ

```
read(N);
readln(a,b,c);
```

Lệnh thứ nhất để nhập một giá trị từ bàn phím và gán giá trị đó cho biến *N*. Lệnh thứ hai dùng để nhập lần lượt ba giá trị từ bàn phím và gán các giá trị đó tương ứng cho ba biến *a*, *b* và *c*.

Khi nhập giá trị cho nhiều biến, những giá trị này được gõ cách nhau bởi ít nhất một dấu cách hoặc kí tự xuống dòng (nhấn phím **Enter**). Các giá trị ứng với biến nguyên phải được biểu diễn dưới dạng số nguyên (không có dấu chấm thập phân). Các giá trị ứng với biến thực có thể được nhập dưới dạng số nguyên, số thực dạng thông thường hoặc số thực dạng dấu phẩy động.

Ví dụ, để nhập các giá trị 1, −5 và 6 cho các biến thực *a*, *b*, *c* trong thủ tục thứ hai trong ví dụ trên, có thể gõ:

`1 -5 6` rồi nhấn phím **Enter**

hoặc

`1.0 -5` rồi nhấn phím **Enter**
`6` rồi nhấn phím **Enter**.

2. Đưa dữ liệu ra màn hình

Để đưa dữ liệu ra màn hình, Pascal cung cấp thủ tục chuẩn:

`write(`*<danh sách kết quả ra>*`);`

hoặc

`writeln(`*<danh sách kết quả ra>*`);`

trong đó, *danh sách kết quả ra* có thể là *tên biến đơn, biểu thức* hoặc *hằng*. Các hằng xâu thường được dùng để tách các kết quả hoặc đưa ra chú thích. Các thành phần trong kết quả ra được viết cách nhau bởi dấu phẩy.

Với thủ tục *write*, sau khi đưa các kết quả ra màn hình, con trỏ không chuyển xuống dòng tiếp theo. Với thủ tục *writeln*, sau khi đưa thông tin ra màn hình, con trỏ sẽ chuyển xuống đầu dòng tiếp theo.

Ví dụ

Để nhập giá trị cho biến *M* từ bàn phím, người ta thường dùng cặp thủ tục:

```
write('Hay nhap gia tri M: ');
readln(M);
```

Khi thực hiện các lệnh này, trên màn hình xuất hiện dòng thông báo:

```
Hay nhap gia tri M:
```

và con trỏ sẽ ở vị trí tiếp theo trên dòng, chờ ta gõ giá trị của *M*.

Để chương trình được sử dụng một cách tiện lợi, khi nhập giá trị từ bàn phím cho biến, ta nên có thêm xâu kí tự nhắc nhở việc nhập giá trị cho biến nào, kiểu dữ liệu gì,... Ví dụ, khi cần nhập một số nguyên dương N ($N \leq 100$) từ bàn phím, ta có thể sử dụng cặp thủ tục sau:

```
write('Nhap so nguyen duong N <= 100: ');
readln(N);
```

Ví dụ

Sau đây là một chương trình hoàn chỉnh có sử dụng các thủ tục vào và ra.

```
program Vidu;
var N: byte;
begin
   write(' Lop ban co bao nhieu nguoi? ');
   readln(N);
   writeln(' Vay la ban co ', N-1,' nguoi ban trong lop.');
   write('Go ENTER de ket thuc chuong trinh.');
   readln
end.
```

Thủ tục *readln* cuối cùng dùng để tạm dừng thực hiện chương trình cho người dùng có thể quan sát kết quả của chương trình đưa ra trên màn hình. Muốn chương trình chạy tiếp cần nhấn phím **Enter**.

Chú ý:
- Các thủ tục *readln* và *writeln* có thể không có tham số.
- Trong thủ tục *write* hoặc *writeln*, sau mỗi kết quả ra (biến, hằng, biểu thức) có thể có *quy cách ra*. Quy cách ra có dạng:

– Đối với kết quả thực:

: *<độ rộng>*: *<số chữ số phần thập phân>*

– Đối với các kết quả khác:

: *<độ rộng>*

trong đó, *độ rộng* và *số chữ số phần thập phân* là các hằng nguyên không âm.

Ví dụ

```
writeln(N:5,x:6:2);
write(i:3,j:4,a+b:8:3);
```

```
␣␣␣ 36 ␣ 24.00
425 ␣␣ 56 ␣␣ 23.200
```

Minh hoạ với N = 36, x = 24, i = 425, j = 56 và a + b = 23.2

Trong thủ tục thứ nhất, 5 vị trí kể từ vị trí con trỏ hiện thời được dành để đưa ra giá trị *N*. Nếu *N* có giá trị nguyên dưới 5 chữ số hoặc giá trị âm dưới 4 chữ số thì những vị trí đầu sẽ được điền dấu cách. Tiếp theo là 6 vị trí được dành để đưa *x* ra, trong đó 2 vị trí dành để đưa ra phần thập phân. Do phần nguyên và phần thập phân được cách nhau bởi dấu chấm nên còn lại 3 vị trí cho phần nguyên.

Trong thủ tục thứ hai, *i* được đưa ra trên 3 vị trí, *j* được đưa ra trên 4 vị trí và kết quả *a* + *b* được đưa ra trên 8 vị trí, trong đó có 3 vị trí dành cho phần thập phân.

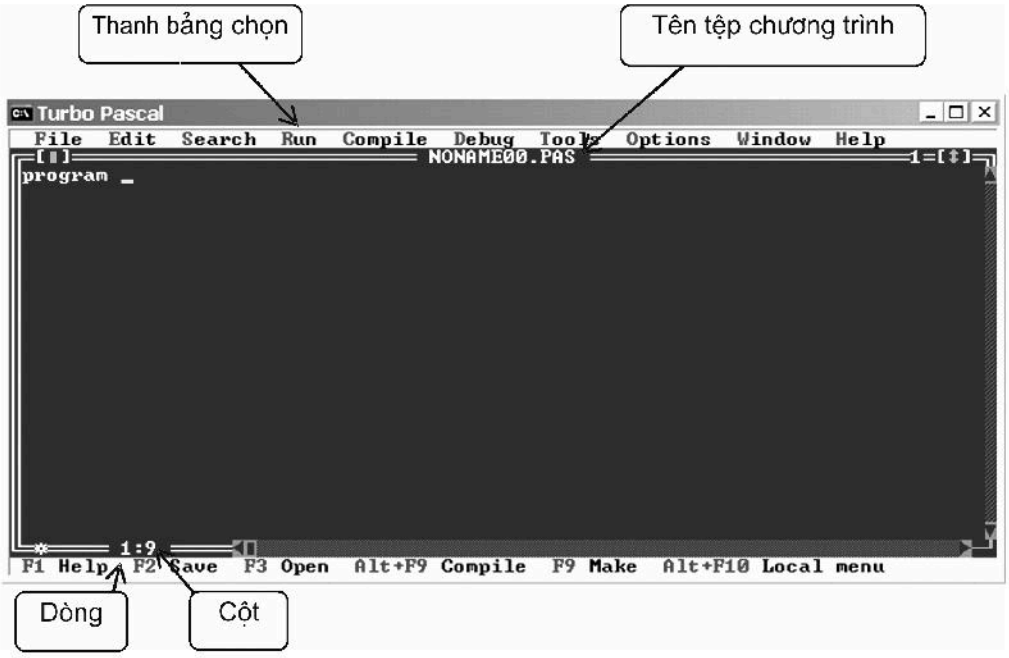
§8. SOẠN THẢO, DỊCH, THỰC HIỆN VÀ HIỆU CHỈNH CHƯƠNG TRÌNH

Để có thể thực hiện chương trình được viết bằng một ngôn ngữ lập trình, ta cần soạn thảo, sử dụng chương trình dịch để dịch chương trình đó sang ngôn ngữ máy. Các hệ thống lập trình cụ thể thường cung cấp phần mềm phục vụ cho việc soạn thảo, dịch và hiệu chỉnh chương trình.

Với ngôn ngữ Pascal, người ta thường dùng các phần mềm Turbo Pascal 7.0 hay Free Pascal. Free Pascal cho phép tận dụng nhiều hơn khả năng của hệ thống. Tuy nhiên, cách làm việc với Turbo Pascal và Free Pascal là tương tự nên dưới đây chỉ giới thiệu cách làm việc với Turbo Pascal.

Trong khuôn khổ sách giáo khoa, để thực hiện các ví dụ và bài thực hành, trong máy tính cần có các tệp: **turbo.exe**, **turbo.tpl**, **graph.tpu**, **egavga.bgi**.

Màn hình làm việc của Pascal có dạng như hình 1 dưới đây.



Hình 1. *Màn hình làm việc của Pascal*

Dòng thứ hai của màn hình được gọi là thanh bảng chọn, mỗi mục trong bảng chọn tương ứng với một nhóm việc ta có thể lựa chọn, hai số ở phía dưới của màn hình ngăn cách nhau bằng dấu hai chấm (:) cho ta biết con trỏ soạn thảo đang ở dòng nào và cột nào trên màn hình.

Dưới đây là một số thao tác và phím tắt thường sử dụng để soạn thảo và thực hiện một chương trình viết bằng Pascal.

- *Soạn thảo:* Gõ nội dung của chương trình gồm phần khai báo và các lệnh trong thân chương trình. Về cơ bản, việc soạn thảo chương trình tương tự như soạn thảo văn bản. Lưu chương trình vào đĩa bằng cách nhấn phím **F2**, nhập tên tệp rồi nhấn phím **Enter** (phần mở rộng của tệp ngầm định là **.pas**).
- *Biên dịch chương trình:* Nhấn tổ hợp phím **Alt+F9**. Nếu chương trình có lỗi cú pháp, phần mềm sẽ hiển thị một thông báo. Cần phải sửa lỗi nếu có, lưu lại chương trình rồi tiến hành biên dịch lại cho tới khi không còn lỗi.
- *Chạy chương trình:* Nhấn tổ hợp phím **Ctrl+F9**.
- *Đóng cửa sổ chương trình:* Nhấn tổ hợp phím **Alt+F3**.
- *Thoát khỏi phần mềm:* Nhấn tổ hợp phím **Alt+X**.

TÓM TẮT

- Dữ liệu của bài toán được biểu diễn thông qua biến trong chương trình theo các quy tắc của ngôn ngữ lập trình cụ thể.
- Kiểu dữ liệu của mọi ngôn ngữ lập trình chỉ cho phép mô tả các đại lượng rời rạc và hữu hạn.
- Một chương trình thường có hai phần: Phần khai báo và phần thân chương trình. Phần khai báo có thể có hoặc không.
- Kiểu dữ liệu chuẩn: Kiểu nguyên, kiểu thực, kiểu kí tự, kiểu lôgic.
- Các biến đều phải được khai báo và mỗi biến chỉ khai báo một lần.
- Các phép toán: số học, quan hệ và lôgic.
- Các loại biểu thức: số học, quan hệ và lôgic.
- Các ngôn ngữ lập trình có:
  - Lệnh gán dùng để gán giá trị của biểu thức cho biến.
  - Các thủ tục chuẩn dùng để đưa dữ liệu vào và ra.

Bài tập và thực hành 1

1. Mục đích, yêu cầu

- *Giới thiệu một chương trình Pascal hoàn chỉnh đơn giản;*
- *Làm quen với một số dịch vụ cơ bản của Turbo Pascal hoặc Free Pascal trong việc soạn thảo, lưu trữ, dịch và thực hiện chương trình.*

2. Nội dung

a) Gõ chương trình sau:

```
program  Giai_PTB2;
uses crt;
var a, b, c, D: real;
   x1, x2: real;
begin
   clrscr;
   write('a, b, c: ');
   readln(a,b,c);
   D:= b*b - 4*a*c;
   x1:= (-b - sqrt(D))/(2*a);
   x2:= -b/a - x1;
   write('x1 = ', x1:6:2,' x2 = ',x2:6:2);
   readln
end.
```

Chú ý: – Dấu chấm phẩy (;) dùng để ngăn cách các khai báo và các câu lệnh. Có thể bỏ qua dấu chấm phẩy sau câu lệnh trước từ khoá **end**.
– Sau từ khoá **end** cuối chương trình phải đặt dấu chấm.

b) Nhấn phím **F2** và lưu chương trình với tên là **PTB2.PAS** lên đĩa.

c) Nhấn tổ hợp phím **Alt+F9** để dịch và sửa lỗi cú pháp (nếu có).

d) Nhấn tổ hợp phím **Ctrl+F9** để thực hiện chương trình. Nhập các giá trị 1; –3 và 2. Quan sát kết quả hiển thị trên màn hình ($x1 = 1.00$ $x2 = 2.00$). Nhấn phím **Enter** để quay lại màn hình soạn thảo.

e) Nhấn tổ hợp phím **Ctrl+F9** rồi nhập các giá trị 1; 0; –2.

Quan sát kết quả hiển thị trên màn hình ($x1 = -1.41$ $x2 = 1.41$).

f) Chỉnh sửa chương trình trên để có chương trình không dùng biến trung gian *D*. Thực hiện chương trình đã sửa với các bộ dữ liệu trên.

g) Sửa lại chương trình nhận được ở câu c) bằng cách thay đổi công thức tính $x2$ (có hai cách để tính $x2$).

h) Thực hiện chương trình đã sửa với bộ dữ liệu 1; −5; 6. Quan sát kết quả trên màn hình ($x1 = 2.00$ $x2 = 3.00$).

i) Thực hiện chương trình với bộ dữ liệu 1; 1; 1 và quan sát kết quả trên màn hình.

CÂU HỎI VÀ BÀI TẬP

1. Hãy cho biết sự khác nhau giữa hằng có đặt tên và biến.

2. Tại sao phải khai báo biến?

3. Trong Pascal, nếu một biến chỉ nhận giá trị nguyên trong phạm vi từ 10 đến 25532 thì biến đó có thể được khai báo bằng các kiểu dữ liệu nào?

4. Biến *P* có thể nhận các giá trị 5; 10; 15; 20; 30; 60; 90 và biến *X* có thể nhận các giá trị 0,1; 0,2; 0,3; 0,4; 0,5. Khai báo nào trong các khai báo sau là đúng?

a) `var X,P: byte;`

b) `var P,X: real;`

c)
```
var P: real;
    X: byte;
```

d)
```
var X:real;
    P:byte;
```

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

a) `var S: integer;`

b) `var S: real;`

c) `var S: word;`

d) `var S: longint;`

e) `var S: boolean;`

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

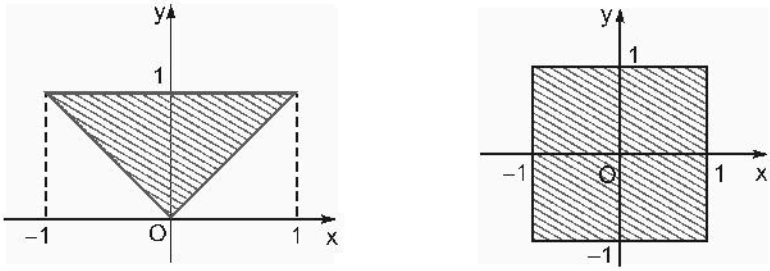
$$(1+z)\frac{x+\frac{y}{z}}{a-\frac{1}{1+x^3}}.$$

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

a) `a/b*2;` b) `a*b*c/2;`

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

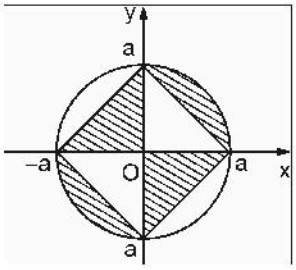
8. Hãy viết biểu thức lôgic cho kết quả *true* khi toạ độ (*x, y*) là điểm nằm trong vùng gạch chéo kể cả biên của các hình 2.*a* và 2.*b*.



a) *b)*

Hình 2. *Các miền cần xác định*

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



Hình 3

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

Chương III

CẤU TRÚC RẼ NHÁNH VÀ LẶP

§9. CẤU TRÚC RẼ NHÁNH

1. Rẽ nhánh

Thường ngày, có rất nhiều việc chỉ được thực hiện khi một điều kiện cụ thể nào đó được thoả mãn.

Ví dụ, Châu và Ngọc thường cùng nhau chuẩn bị các bài thực hành môn Tin học.

Một lần Châu hẹn với Ngọc: "*Chiều mai nếu trời không mưa thì Châu sẽ đến nhà Ngọc*".

Một lần khác, Ngọc nói với Châu: "*Chiều mai nếu trời không mưa thì Ngọc sẽ đến nhà Châu, nếu mưa thì sẽ gọi điện cho Châu để trao đổi*".

Câu nói của Châu cho ta biết một việc làm cụ thể (Châu đến nhà Ngọc) sẽ được thực hiện nếu một điều kiện cụ thể (trời không mưa) thoả mãn. Ngoài ra không đề cập đến việc gì sẽ xảy ra nếu điều kiện đó không thoả mãn (trời mưa).

Ta nói cách diễn đạt như vậy thuộc dạng thiếu:

Nếu... thì...

Câu nói của Ngọc khẳng định một trong hai việc cụ thể (Ngọc đến nhà Châu hay Ngọc gọi điện cho Châu) chắc chắn sẽ xảy ra. Tuy nhiên, việc nào trong hai việc sẽ được thực hiện thì tuỳ thuộc vào điều kiện cụ thể (trời không mưa) thoả mãn hay không.

Ta nói cách diễn đạt như vậy thuộc dạng đủ:

Nếu... thì..., nếu không thì...

Từ đó có thể thấy, trong nhiều thuật toán, các thao tác tiếp theo sẽ phụ thuộc vào kết quả nhận được từ các bước trước đó.

Cấu trúc dùng để mô tả các mệnh đề có dạng như trên được gọi là cấu trúc rẽ nhánh thiếu và đủ.

Ví dụ, để giải phương trình bậc hai:

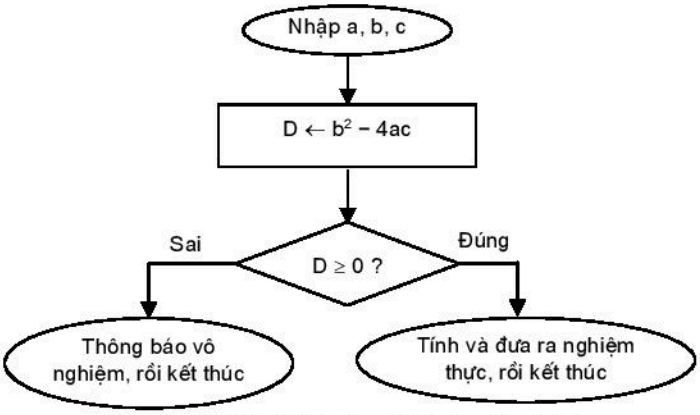
$$ax^2 + bx + c = 0, (a \neq 0)$$

trước tiên, ta tính biệt số delta $D = b^2 - 4ac$.

Nếu D không âm, ta sẽ đưa ra các nghiệm. Trong trường hợp ngược lại, ta phải thông báo là phương trình vô nghiệm.

Như vậy, sau khi tính D, tuỳ thuộc vào giá trị của D, một trong hai thao tác sẽ được thực hiện (h. 4).

Mọi ngôn ngữ lập trình đều có các câu lệnh để mô tả cấu trúc rẽ nhánh.



Hình 4. *Sơ đồ thể hiện cấu trúc rẽ nhánh*

2. Câu lệnh if-then

Để mô tả cấu trúc rẽ nhánh, Pascal dùng câu lệnh *if-then*. Tương ứng với hai dạng thiếu và đủ nói ở trên, Pascal có hai dạng câu lệnh *if-then*:

a) Dạng thiếu

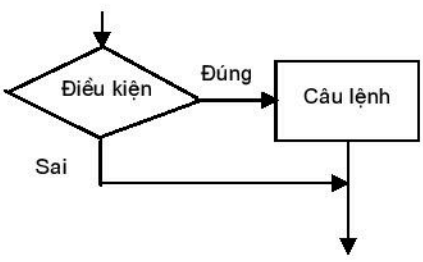
`if` <*điều kiện*> `then` <*câu lệnh* >`;`

b) Dạng đủ

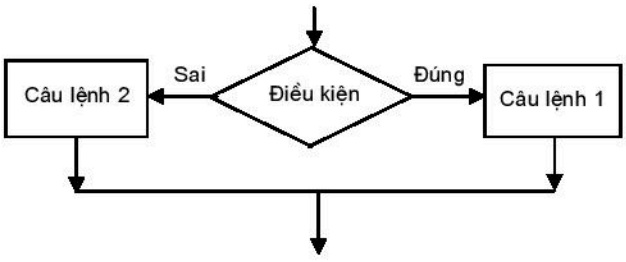
`if` <*điều kiện*> `then` <*câu lệnh 1*> `else` <*câu lệnh 2*>`;`

trong đó:

- *Điều kiện* là biểu thức lôgic.
- *Câu lệnh, câu lệnh 1, câu lệnh 2* là một câu lệnh của Pascal.



Hình 5



Hình 6

Ở dạng thiếu: *điều kiện* sẽ được tính và kiểm tra. Nếu *điều kiện* đúng (có giá trị *true*) thì *câu lệnh* sẽ được thực hiện, ngược lại thì *câu lệnh* sẽ bị bỏ qua (h. 5).

Ở dạng đủ: *điều kiện* cũng được tính và kiểm tra. Nếu *điều kiện* đúng thì *câu lệnh 1* sẽ được thực hiện, ngược lại thì *câu lệnh 2* sẽ được thực hiện (h. 6).

Ví dụ 1

```
if D < 0 then writeln('Phuong trinh vo nghiem.');
```

Ví dụ 2

```
if a mod 3 = 0 then write('a chia het cho 3')
   else write('a khong chia het cho 3');
```

Ví dụ 3. Để tìm số lớn nhất *max* trong hai số *a* và *b*, có thể thực hiện bằng hai cách sau:

– Dùng câu lệnh gán *max*:= *a* và lệnh *if-then* dạng thiếu:

```
if b > a then max:=b;
```

– Dùng một lệnh *if-then* dạng đủ:

```
if b > a then max:=b else max:=a;
```

3. Câu lệnh ghép

Theo cú pháp, sau một số từ khoá (như *then* hoặc *else*) phải là một câu lệnh. Nhưng trong nhiều trường hợp, các thao tác sau những tên dành riêng đó khá phức tạp, đòi hỏi không phải chỉ một mà là nhiều câu lệnh để mô tả. Trong các trường hợp như vậy, ngôn ngữ lập trình cho phép gộp một dãy câu lệnh thành một *câu lệnh ghép* (hay câu lệnh hợp thành). Chẳng hạn, câu lệnh ghép của Pascal có dạng:

```
begin
   <các câu lệnh> ;
end;
```

Câu lệnh, *câu lệnh 1*, *câu lệnh 2* trong các câu lệnh *if-then* ở mục trên có thể là câu lệnh ghép.

Thuật ngữ *câu lệnh* được hiểu chung cho cả câu lệnh đơn và câu lệnh ghép.

Ví dụ

```
if D < 0 then writeln('Phuong trinh vo nghiem.')
  else
     begin
        x1:= (-b - sqrt(b*b - 4*a*c))/(2*a);
        x2:= -b/a-x1;
     end;
```

4. Một số ví dụ

Ví dụ 1. Tìm nghiệm thực của phương trình bậc hai:

$$ax^2 + bx + c = 0 \text{ , với } a \neq 0.$$

Input: Các hệ số a, b, c nhập từ bàn phím.

Output: Đưa ra màn hình các nghiệm thực hoặc thông báo "*Phuong trinh vo nghiem*".

```
program Giai_PTB2;
uses crt;
var a,b,c: real;
   D, x1, x2: real;
begin
   clrscr;
   write(' a, b, c: ');
   readln(a, b, c);
   D:= b*b - 4*a*c;
   if D < 0 then writeln('Phuong trinh vo nghiem.')
      else
      begin
         x1:= (-b - sqrt(D))/(2*a);
         x2:= -b/a - x1;
         writeln(' x1 = ', x1:8:3,' x2 = ', x2:8:3);
      end;
   readln
end.
```

Ví dụ 2

Tìm số ngày của năm N, biết rằng năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100. Ví dụ, các năm 2000, 2004 là năm nhuận và có số ngày là 366, các năm 1900, 1945 không phải là năm nhuận và có số ngày là 365.

Input: N nhập từ bàn phím.

Output: Đưa số ngày của năm N ra màn hình.

```
program Nam_nhuan;
uses crt;
var N, SN: integer;
begin
   clrscr;
   write('Nam: '); readln(N);
   if (N mod 400 = 0)or((N mod 4 = 0)and(N mod 100 <> 0))
      then SN:= 366 else SN:= 365;
   writeln(' So ngay cua nam ', N,' la ', SN);
   readln
end.
```

§10. CẤU TRÚC LẶP

1. Lặp

Với a là số nguyên và $a > 2$, xét các bài toán sau đây:

Bài toán 1. Tính và đưa kết quả ra màn hình tổng

$$S = \frac{1}{a} + \frac{1}{a+1} + \frac{1}{a+2} + \ldots + \frac{1}{a+100}.$$

Bài toán 2. Tính và đưa kết quả ra màn hình tổng

$$S = \frac{1}{a} + \frac{1}{a+1} + \frac{1}{a+2} + \ldots + \frac{1}{a+N} + \ldots$$

cho đến khi $\frac{1}{a+N} < 0{,}0001$.

Với cả hai bài toán, dễ thấy cách để tính tổng S có nhiều điểm tương tự:

- Xuất phát, S được gán giá trị $\frac{1}{a}$;
- Tiếp theo, cộng vào tổng S một giá trị $\frac{1}{a+N}$ với $N = 1, 2, 3, 4, 5,\ldots$

  Việc cộng này được lặp lại một số lần.

Đối với bài toán 1, số lần lặp là 100 và việc cộng vào tổng S sẽ kết thúc khi đã thực hiện việc cộng 100 lần.

Đối với bài toán 2, số lần lặp chưa biết trước nhưng việc cộng vào tổng S sẽ kết thúc khi điều kiện $\frac{1}{a+N} < 0{,}0001$ được thoả mãn.

Nói chung, trong một số thuật toán có những thao tác phải thực hiện lặp đi lặp lại một số lần. Một trong các đặc trưng của máy tính là có khả năng thực hiện hiệu quả các thao tác lặp. Cấu trúc lặp mô tả thao tác lặp và có hai dạng là *lặp với số lần biết trước* và *lặp với số lần chưa biết trước*.

Các ngôn ngữ lập trình đều có các câu lệnh để mô tả cấu trúc lặp.

2. Lặp với số lần biết trước và câu lệnh for-do

Có hai thuật toán *Tong_1a* và *Tong_1b* để giải bài toán 1 như sau:

Thuật toán Tong_1a

Bước 1. $S \leftarrow 1/a$; $N \leftarrow 0$; {Khởi tạo S và N}

Bước 2. $N \leftarrow N + 1$;

Bước 3. Nếu $N > 100$ thì chuyển đến bước 5;

Bước 4. $S \leftarrow S + 1/(a + N)$ rồi quay lại bước 2;

Bước 5. Đưa S ra màn hình, rồi kết thúc.

Thuật toán Tong_1b

Bước 1. $S \leftarrow 1/a$; $N \leftarrow 101$; {Khởi tạo S và N}

Bước 2. $N \leftarrow N - 1$;

Bước 3. Nếu $N < 1$ thì chuyển đến bước 5;

Bước 4. $S \leftarrow S + 1/(a + N)$ rồi quay lại bước 2;

Bước 5. Đưa S ra màn hình rồi kết thúc.

Lưu ý, số lần lặp của cả hai thuật toán trên là biết trước và như nhau (100 lần).

Trong thuật toán *Tong_1a*, giá trị N khi bắt đầu tham gia vòng lặp là 1 và sau mỗi lần lặp N tăng lên 1 cho đến khi $N > 100$ ($N = 101$) thì kết thúc lặp (thực hiện đủ 100 lần). Trong thuật toán *Tong_1b*, giá trị N bắt đầu tham gia vòng lặp là 100 và sau mỗi lần lặp N giảm đi 1 cho đến khi $N < 1$ ($N = 0$) thì kết thúc lặp (thực hiện đủ 100 lần). Ta nói cách lặp trong thuật toán *Tong_1a* là dạng tiến và trong thuật toán *Tong_1b* là dạng lùi.

Để mô tả cấu trúc lặp với số lần biết trước, Pascal dùng câu lệnh lặp *for-do* với hai dạng tiến và lùi như sau:

- *Dạng lặp tiến:*

  `for` <*biến đếm*>`:=` <*giá trị đầu*> `to` <*giá trị cuối*> `do` <*câu lệnh* >`;`

- *Dạng lặp lùi:*

  `for` <*biến đếm*>`:=` <*giá trị cuối*> `downto` <*giá trị đầu* > `do` <*câu lệnh*>`;`

  trong đó:

– *Biến đếm* là biến đơn, thường có kiểu nguyên.

– *Giá trị đầu, giá trị cuối* là các biểu thức cùng kiểu với biến đếm và *giá trị đầu* phải nhỏ hơn hoặc bằng *giá trị cuối*. Nếu *giá trị đầu* lớn hơn *giá trị cuối* thì vòng lặp không được thực hiện.

Hoạt động của lệnh *for-do*:

– Ở dạng *lặp tiến*, câu lệnh viết sau từ khoá *do* được thực hiện tuần tự, với *biến đếm* lần lượt nhận các giá trị liên tiếp tăng từ *giá trị đầu* đến *giá trị cuối*.

– Ở dạng *lặp lùi*, câu lệnh viết sau từ khoá *do* được thực hiện tuần tự, với *biến đếm* lần lượt nhận các giá trị liên tiếp giảm từ *giá trị cuối* đến *giá trị đầu*.

Chú ý: Giá trị của *biến đếm* được điều chỉnh tự động, vì vậy câu lệnh viết sau ***do*** không được thay đổi giá trị *biến đếm*.

Ví dụ 1. Sau đây là hai chương trình cài đặt các thuật toán *Tong_1a* và *Tong_1b*.

```
program Tong_1a;
uses crt;
var S: real;
   a, N: integer;
begin
   clrscr;
   write('Hay nhap gia tri a vao!');
   readln(a);
   S:=1.0/a;                          {Buoc 1}
   for N:= 1 to 100 do                {Buoc 2, Buoc 3}
       S:= S+1.0/(a+N);               {Buoc 4}
   writeln('Tong S la: ', S:8:4);     {Buoc 5}
   readln
end.
program Tong_1b;
uses crt;
var S: real;
   a, N: integer;
begin
   clrscr;
   write ('Hay nhap gia tri a vao!');
   readln(a);
```

```
    S:=1.0/a;                          {Buoc 1}
    for N:= 100 downto 1 do            {Buoc 2 va Buoc 3}
        S:= S+1.0/(a+N);               {Buoc 4}
    writeln('Tong S la: ', S:8:4);     {Buoc 5}
    readln
end.
```

Ví dụ 2. Chương trình sau thực hiện việc nhập từ bàn phím hai số nguyên dương M và N ($M < N$), tính và đưa ra màn hình tổng các số chia hết cho 3 hoặc 5 trong phạm vi từ M đến N.

```
program Vi_du_2;
uses crt;
var M, N, I: integer;
    T: longint;
begin
   clrscr;
   writeln('Nhap so M nho hon N');
   write('M = ');readln(M);
   write('N = ');readln(N);
   T:= 0;
   for I:= M to N do
      if (I mod 3 = 0) or (I mod 5 = 0) then
         T:=T+I;
   writeln('KET QUA: ', T);
   readln
end.
```

3. Lặp với số lần chưa biết trước và câu lệnh while-do

Có thể xây dựng thuật toán *Tong_2* như sau để giải bài toán 2.

Thuật toán Tong_2

Bước 1. $S \leftarrow 1/a$; $N \leftarrow 0$; {Khởi tạo S và N}

Bước 2. Nếu $1/(a + N) < 0{,}0001$ thì chuyển đến bước 5;

Bước 3. $N \leftarrow N + 1$;

Bước 4. $S \leftarrow S + 1/(a + N)$ rồi quay lại bước 2;

Bước 5. Đưa S ra màn hình, rồi kết thúc.

Như vậy, việc lặp với số lần chưa biết trước sẽ chỉ kết thúc khi một điều kiện cho trước được thoả mãn.

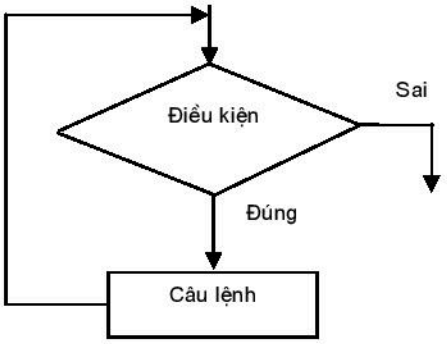
Để mô tả cấu trúc lặp như vậy, Pascal dùng câu lệnh *while-do* có dạng:

while <*điều kiện*> **do**<*câu lệnh* >;

trong đó:

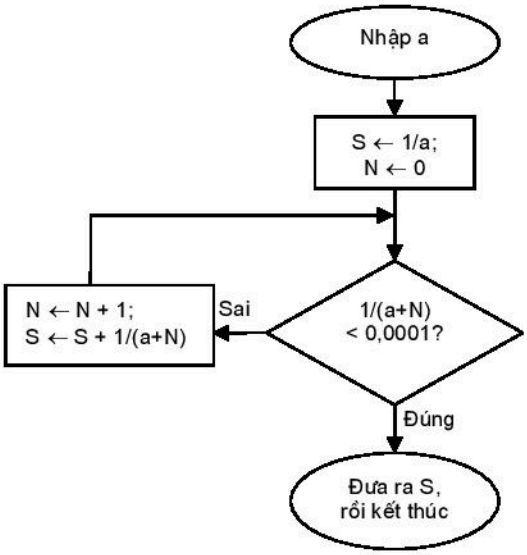
- *Điều kiện* là biểu thức lôgic;
- *Câu lệnh* là một câu lệnh đơn hoặc ghép.

Việc thực hiện lệnh *while-do* được thể hiện trên sơ đồ ở hình 7.



Hình 7. *Sơ đồ lặp với số lần lặp chưa biết trước*

Ví dụ 1. Sau đây là sơ đồ khối và chương trình cài đặt thuật toán *Tong_2*.



Hình 8. *Sơ đồ khối của thuật toán Tong_2*

```
program Tong_2;
uses crt;
var    S: real;
       a, N: integer;
begin
   write ('Hay nhap gia tri a vao!');
   readln(a);
   S:= 1.0/a; N:= 0;                        {Buoc 1}
     while not (1/(a+N)<0.0001) do          {Buoc 2}
         begin
             N:= N+1;                       {Buoc 3}
             S:= S+1.0/(a+N);               {Buoc 4}
         end;
   writeln('Tong S la: ', S:8:4);           {Buoc 5}
   readln
end.
```

Ví dụ 2. Tìm ước chung lớn nhất (ƯCLN) của hai số nguyên dương *M* và *N*.

Có nhiều thuật toán khác nhau tìm ƯCLN của *M* và *N*. Sau đây là một thuật toán tìm ƯCLN.

Thuật toán

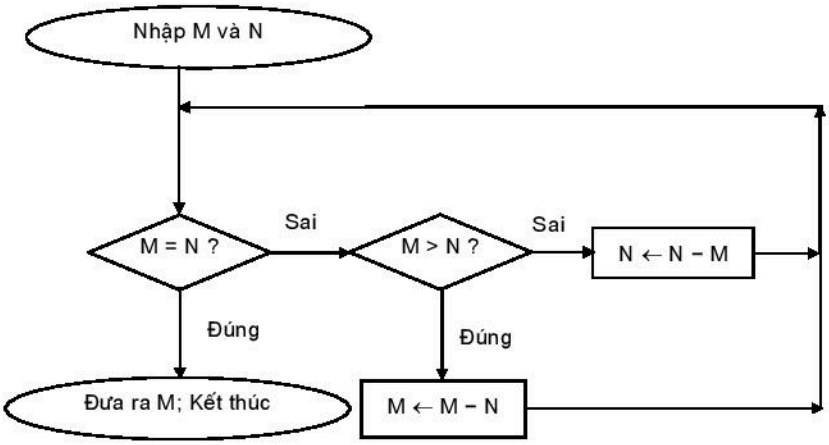
Bước 1. Nhập *M*, *N*;

Bước 2. Nếu *M* = *N* thì lấy giá trị chung này làm ƯCLN rồi chuyển đến bước 5;

Bước 3. Nếu *M* > *N* thì *M* ← *M* − *N* ngược lại *N* ← *N* − *M*;

Bước 4. Quay lại bước 2;

Bước 5. Đưa ra kết quả ƯCLN rồi kết thúc.



Hình 9. *Sơ đồ khối của thuật toán tìm ước chung lớn nhất*

Chương trình sau thể hiện thuật toán tìm ước chung lớn nhất.

```
program UCLN;
uses crt;
var M,N:integer;
begin
   clrscr;
   write('M, N = ');
   readln(M,N);
   while M <> N do
      if M > N then M:= M-N else N:= N-M;
   writeln('UCLN = ', M);
   readln
end.
```

Chú ý: Các câu lệnh trong vòng lặp thường được lặp lại nhiều lần, vì vậy để tăng hiệu quả của chương trình thì những thao tác không cần lặp lại nên đưa ra ngoài vòng lặp.

TÓM TẮT

- Các ngôn ngữ lập trình đều có câu lệnh thể hiện cấu trúc rẽ nhánh và cấu trúc lặp.
- Câu lệnh rẽ nhánh có hai dạng:
  - Dạng thiếu;
  - Dạng đủ.
- Có thể gộp dãy câu lệnh thành câu lệnh ghép.
- Các câu lệnh mô tả cấu trúc lặp:
  - Lặp với số lần biết trước;
  - Lặp với số lần chưa biết trước.

Định lí Bohn Jacopini (Bon Ja-co-pi-ni): Mọi quá trình tính toán đều có thể mô tả và thực hiện dựa trên ba cấu trúc cơ bản là cấu trúc tuần tự, cấu trúc rẽ nhánh và cấu trúc lặp.

Bài tập và thực hành 2

1. Mục đích, yêu cầu

- *Xây dựng chương trình có sử dụng cấu trúc rẽ nhánh;*
- *Làm quen với việc hiệu chỉnh chương trình.*

2. Nội dung

Bài toán. *Bộ số Pi-ta-go*

Biết rằng bộ ba số nguyên dương a, b, c được gọi là bộ số Pi-ta-go nếu tổng các bình phương của hai số bằng bình phương của số còn lại. Viết chương trình nhập từ bàn phím ba số nguyên dương a, b, c và kiểm tra xem chúng có là bộ số Pi-ta-go hay không.

Ý tưởng: Kiểm tra xem có đẳng thức nào trong ba đẳng thức sau đây xảy ra hay không:

$$a^2 = b^2 + c^2$$
$$b^2 = a^2 + c^2$$
$$c^2 = a^2 + b^2.$$

Những công việc cần thực hiện:

a) Gõ chương trình sau:

```
program Pi_ta_go;
uses crt;
var a, b, c: integer;
    a2, b2, c2: longint;
begin
   clrscr;
   write('a, b, c: ');
   readln(a, b, c);
   a2:= a;
   b2:= b;
   c2:= c;
   a2:= a2*a;
```

```
    b2:= b2*b;
    c2:= c2*c;
    if (a2 = b2 + c2) or (b2 = a2 + c2) or (c2 = a2 + b2)
       then writeln('Ba so da nhap la bo so Pi-ta-go')
       else writeln('Ba so da nhap khong la bo so Pi-ta-go');
    readln
end.
```

Chú ý: Trước else không có dấu chấm phẩy (;).

b) Lưu chương trình với tên **PITAGO** lên đĩa.

c) Nhấn phím **F7** để thực hiện từng câu lệnh chương trình, nhập các giá trị $a = 3, b = 4, c = 5$.

d) Vào bảng chọn **Debug** mở cửa sổ hiệu chỉnh để xem giá trị $a2, b2, c2$.

e) Nhấn phím **F7** để thực hiện các câu lệnh tính những giá trị nói trên, so sánh với kết quả $a2 = 9, b2 = 16, c2 = 25$.

f) Quan sát quá trình rẽ nhánh.

g) Lặp lại các bước trên với bộ dữ liệu $a = 700, b = 1000, c = 800$.

h) Nếu thay dãy lệnh

```
a2:= a;
b2:= b;
c2:= c;
a2:= a2*a;
b2:= b2*b;
c2:= c2*c;
```

bằng dãy lệnh

```
a2:= a*a;
b2:= b*b;
c2:= c*c;
```

thì kết quả có gì thay đổi với bộ dữ liệu cho ở câu *g*?

CÂU HỎI VÀ BÀI TẬP

1. Hãy cho biết sự giống và khác nhau của hai dạng câu lệnh **If-then**.

2. Câu lệnh ghép là gì? Tại sao phải có câu lệnh ghép?

3. Có thể dùng câu lệnh **while-do** để thay cho câu lệnh **for-do** được không? Nếu được, hãy thực hiện điều đó với chương trình *Tong_1a*.

4. Viết câu lệnh rẽ nhánh tính:

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

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

5. Lập trình tính:

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

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

Đưa giá trị $e(n)$ ra màn hình.

6. Lập trình để giải bài toán cổ sau:

Vùa gà vùa chó.
Bó lại cho tròn.
Ba mươi sáu con,
Một trăm chân chẵn.
Hỏi có bao nhiêu con mỗi loại?

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

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

Chương IV

KIỂU DỮ LIỆU CÓ CẤU TRÚC

§11. KIỂU MẢNG

Chúng ta chỉ xét hai kiểu mảng thông dụng với nhiều ngôn ngữ lập trình là kiểu mảng một chiều và kiểu mảng hai chiều.

1. Kiểu mảng một chiều

Mảng một chiều là dãy hữu hạn các phần tử cùng kiểu. Mảng được đặt tên và mỗi phần tử của nó có một chỉ số. Để mô tả mảng một chiều cần xác định kiểu của các phần tử và cách đánh số các phần tử của nó.

Để người lập trình có thể xây dựng và sử dụng kiểu mảng một chiều, các ngôn ngữ lập trình có quy tắc, cách thức cho phép xác định:

- Tên kiểu mảng một chiều;
- Số lượng phần tử;
- Kiểu dữ liệu của phần tử;
- Cách khai báo biến mảng;
- Cách tham chiếu đến phần tử.

Ví dụ, xét bài toán nhập vào nhiệt độ (trung bình) của mỗi ngày trong tuần, tính và đưa ra màn hình nhiệt độ trung bình của tuần và số lượng ngày trong tuần có nhiệt độ cao hơn nhiệt độ trung bình của tuần.

Ta có thể dùng bảy biến thực để lưu trữ nhiệt độ của các ngày trong tuần. Chương trình giải bài toán có thể được viết bằng Pascal như sau:

```
program Nhietdo_Tuan;
var t1,t2,t3,t4,t5,t6,t7,tb: real;
    dem: integer;
begin
   writeln('Nhap vao nhiet do cua 7 ngay: ');
   readln(t1,t2,t3,t4,t5,t6,t7);
   tb:= (t1+t2+t3+t4+t5+t6+t7)/7;
   dem:= 0;
   if t1>tb then dem:= dem+1;
```

```
    if t2>tb then dem:= dem+1;
    if t3>tb then dem:= dem+1;
    if t4>tb then dem:= dem+1;
    if t5>tb then dem:= dem+1;
    if t6>tb then dem:= dem+1;
    if t7>tb then dem:= dem+1;
    writeln('Nhiet do trung binh tuan: ',tb:4:2);
    writeln('So ngay nhiet do cao hon trung binh: ',dem);
    readln
end.
```

Khi cần giải bài toán trên với *N* ngày (*N* khá lớn) thì cách làm tương tự không những đòi hỏi một khối lượng khai báo khá lớn, mà đoạn chương trình tính toán cũng khá dài.

Để giải quyết vấn đề đó, ta sử dụng kiểu dữ liệu mảng một chiều để mô tả dữ liệu. Chương trình giải bài toán tổng quát với *N* ngày trong Pascal có thể như sau:

```
program Nhietdo_Nngay;
const Max = 366;    {gia thiet N lon nhat la 366}
type Kmang1= array[1..Max] of real;
var Nhietdo: Kmang1;
    dem, i, N: integer;
    Tong, Trung_binh: real;
begin
    write('Nhap so ngay: '); readln(N);
    Tong:= 0;
    for i:= 1 to N do
        begin
            write('Nhap nhiet do ngay ',i, ': ');
            readln(Nhietdo[i]);
            Tong:= Tong + Nhietdo[i];
        end;
    dem:= 0;
    Trung_binh:= Tong/N;
    for i:= 1 to N do
        if Nhietdo[i]>Trung_binh then dem:= dem+1;
    writeln('Nhiet do trung binh',N,' ngay: ',Trung_binh:8:3);
    writeln('So ngay nhiet do cao hon trung binh: ',dem);
    readln
end.
```

Trong chương trình này đã khai báo biến mảng một chiều (thông qua khai báo kiểu mảng) như sau:

```
Type Kmang1 = array[1..Max] of real;
Var Nhietdo: Kmang1;
```

Khai báo *kiểu mảng* một chiều gồm *Max* số thực.

Khai báo *biến mảng* *Nhietdo* qua kiểu mảng.

a) Khai báo

Tổng quát, khai báo biến mảng một chiều có dạng:

– *Cách 1*. Khai báo *trực tiếp* biến mảng một chiều:

var <*tên biến mảng* >: **array** [*kiểu chỉ số*] **of** <*kiểu phần tử*>;

– *Cách 2*. Khai báo *gián tiếp* biến mảng qua kiểu mảng một chiều:

type <*tên kiểu mảng*> = **array** [*kiểu chỉ số*] **of** <*kiểu phần tử*>;

var <*tên biến mảng* >: <*tên kiểu mảng*>;

trong đó:

- *Kiểu chỉ số* thường là một đoạn số nguyên liên tục có dạng *n*1..*n*2 với *n*1, *n*2 là các hằng hoặc biểu thức nguyên xác định chỉ số đầu và chỉ số cuối (*n*1 ≤ *n*2);
- *Kiểu phần tử* là kiểu của các phần tử mảng.

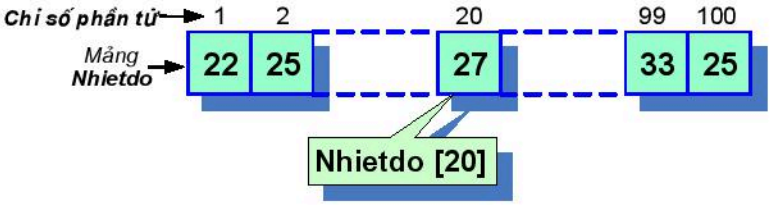
Ví dụ. Các khai báo kiểu mảng một chiều sau đây là hợp lệ:

```
type
   ArrayReal = array[-100..200] of real;
   ArrayBoolean = array[-n+1..n+1] of boolean;
   ArrayInt = array[-100..0] of integer;
```

trong đó *n* là hằng nguyên.

Tham chiếu tới phần tử của mảng một chiều được xác định bởi tên mảng cùng với chỉ số, được viết trong cặp ngoặc [và].

Ví dụ, tham chiếu tới nhiệt độ của ngày thứ 20, trong chương trình trên, được viết là *Nhietdo*[20].



Hình 10. *Minh hoạ mảng một chiều*

b) Một số ví dụ

Ta xét chương trình có sử dụng mảng một chiều cài đặt một số thuật toán giải những bài toán tìm kiếm và sắp xếp.

Ví dụ 1. *Tìm phần tử lớn nhất của dãy số nguyên*

Input: Số nguyên dương N ($N \le 250$) và dãy N số nguyên dương $A_1, A_2,..., A_N$, mỗi số đều không vượt quá 500.

Output: Chỉ số và giá trị của phần tử lớn nhất trong dãy số đã cho (nếu có nhiều phần tử lớn nhất chỉ cần đưa ra một trong số chúng).

Bước 1. Nhập N và dãy $A_1,..., A_N$;
Bước 2. Max $\leftarrow A_1$, $i \leftarrow 2$;
Bước 3. Nếu $i > N$ thì đưa ra giá trị Max rồi kết thúc;
Bước 4.
Bước 4.1. Nếu $A_i >$ Max thì Max $\leftarrow A_i$;
Bước 4.2. $i \leftarrow i + 1$ rồi quay lại bước 3;

Hình 11. *Thuật toán tìm phần tử lớn nhất của dãy số*

Chương trình dưới đây thực hiện việc duyệt tuần tự các phần tử để tìm ra số lớn nhất của dãy số nguyên.

```
program TimMax;
uses crt;
const
   Nmax = 250;
type
   ArrInt = array[1..Nmax] of integer;
var
   N,i, Max, csmax: integer;
   A: ArrInt;
begin
   clrscr;
   write('Nhap so luong phan tu cua day so, N = ');
   readln(N);
   for i:=1 to N do
      begin
         write('Phan tu thu ',i,' = ');
         readln(A[i]);
      end;
```

```
    Max:= A[1]; csmax:=1;
    for i:=2 to N do
        if A[i]> Max then
           begin
              Max:= A[i];
              csmax:= i;
           end;
    writeln('Gia tri cua phan tu Max: ', Max);
    writeln('Chi so cua phan tu Max: ', csmax);
    readln
end.
```

Ví dụ 2. *Sắp xếp dãy số nguyên bằng thuật toán tráo đổi*

Input: Số nguyên dương N ($N \leq 250$) và dãy A gồm N số nguyên dương $A_1, A_2,..., A_N$, mỗi số đều không vượt quá 500.

Output: Dãy số A đã được sắp xếp thành dãy không giảm.

Để giải bài toán này, ta sẽ sử dụng thuật toán tráo đổi như mô tả trong sách giáo khoa Tin học 10.

```
(* Chuong trinh giai bai toan sap xep day so *)
program sapxep;
uses crt;
const Nmax = 250;
type
    ArrInt = array[1..Nmax] of integer;
var
    N,i,j,t: integer;
    A: ArrInt;
begin
    clrscr;
    write('Nhap so luong phan tu cua day so, N = ');
    readln(N);
    for i:=1 to N do (* Nhap cac phan tu *)
       begin
          write('Phan tu thu ',i,' = ');
          readln(A[i]);
       end;
    for j:=N downto 2 do
       for i:=1 to j-1 do
            if A[i]> A[i+1] then
```

```
                begin  (*Trao doi A[i] va A[i+1]*)
                   t:= A[i];
                   A[i]:= A[i+1];
                   A[i+1]:= t;
                end;
   writeln('Day so duoc sap xep la: ');
   for i:=1 to N do write(A[i]: 4);
   readln
end.
```

Ví dụ 3. *Tìm kiếm nhị phân*

Input: Dãy A là dãy tăng gồm N ($N \le 250$) số nguyên dương $A_1, A_2,..., A_N$ và số nguyên k.

Output: Chỉ số i mà $A_i = k$ hoặc thông báo "*Khong tim thay*" nếu không có số hạng nào của dãy A có giá trị bằng k.

Để giải bài toán này, chúng ta sẽ sử dụng thuật toán tìm kiếm nhị phân được trình bày trong sách giáo khoa Tin học 10.

Bước 1. Nhập N, các số hạng $A_1, A_2,..., A_N$ và khoá k;

Bước 2. $Dau \leftarrow 1$, $Cuoi \leftarrow N$;

Bước 3. $Giua \leftarrow \left[\frac{Dau + Cuoi}{2}\right]$;

Bước 4. Nếu $A_{Giua} = k$ thì thông báo chỉ số $Giua$, rồi kết thúc;

Bước 5. Nếu $A_{Giua} > k$ thì đặt $Cuoi = Giua - 1$ rồi chuyển đến bước 7;

Bước 6. $Dau \leftarrow Giua + 1$;

Bước 7. Nếu $Dau > Cuoi$ thì thông báo dãy A không có số hạng có giá trị bằng k, rồi kết thúc;

Bước 8. Quay lại bước 3.

Hình 12. *Thuật toán tìm kiếm nhị phân*

```
(* Chuong trinh cai dat thuat toan tim kiem nhi phan*)
program TK_nhiphan;
uses crt;
const
   Nmax = 250;
type
   ArrInt = array[1..Nmax] of integer;
var
   N, i, k: integer;
   Dau, Cuoi, Giua: integer;
   A: ArrInt;
   Tim_Thay: boolean;
```

```
begin
    clrscr;
    write('Nhap so luong phan tu cua day so, N = ');
    readln(N);
    writeln('Nhap cac phan tu cua day so tang: ');
    for i:=1 to N do
        begin
            write('Phan tu thu ',i,' = ');
            readln(A[i]);
        end;
    write('Nhap gia tri k = ');
    readln(k);
    Dau:= 1; Cuoi:=N; Tim_thay:= false;
    while (Dau <= Cuoi) and not (Tim_Thay) do
        begin
           Giua:= (Dau+Cuoi) div 2;
           if A[Giua] = k then
              Tim_thay:= true
              else
                 if A[Giua]>k then Cuoi:= Giua-1
                    else Dau:= Giua+1;
        end;
    if Tim_thay then writeln('Chi so tim duoc la: ', Giua)
        else  writeln('Khong tim thay');
    readln
end.
```

2. Kiểu mảng hai chiều

Xét bài toán tính và đưa ra màn hình bảng nhân.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90

Bảng nhân

Ta thấy bảng nhân có dạng bảng gồm các giá trị cùng kiểu. Ta có thể biểu diễn bảng nhân bằng kiểu dữ liệu mảng hai chiều.

Mảng hai chiều là bảng các phần tử cùng kiểu.

Nhận xét rằng mỗi hàng của mảng hai chiều có cấu trúc như một mảng một chiều cùng kích thước. Nếu coi mỗi hàng của mảng hai chiều là một phần tử thì ta có thể nói mảng hai chiều là mảng một chiều mà mỗi phần tử là mảng một chiều.

Như vậy, ta cũng có thể mô tả dữ liệu của bảng nhân là kiểu mảng một chiều gồm 9 phần tử, mỗi phần tử lại là mảng một chiều có 10 phần tử, mỗi phần tử là một số nguyên.

Tương tự như với kiểu mảng một chiều, với kiểu mảng hai chiều, các ngôn ngữ lập trình cũng có các quy tắc, cách thức cho phép xác định:

- Tên kiểu mảng hai chiều;
- Số lượng phần tử của mỗi chiều;
- Kiểu dữ liệu của phần tử;
- Cách khai báo biến;
- Cách tham chiếu đến phần tử.

Ví dụ, biến mảng hai chiều *B* lưu trữ bảng nhân có thể được khai báo trong Pascal như sau:

```
var B: array[1..9] of array[1..10] of integer;
```

hoặc có thể khai báo ngắn gọn:

```
var B: array[1..9,1..10] of integer;
```

a) Khai báo

Tổng quát, khai báo biến mảng hai chiều trong Pascal như sau:

– *Cách 1*. Khai báo trực tiếp biến mảng hai chiều:

var <*tên biến mảng*>**: array[***kiểu chỉ số hàng*, *kiểu chỉ số cột***]**
of <*kiểu phần tử*>;

– *Cách 2*. Khai báo gián tiếp biến mảng qua kiểu mảng hai chiều:

type <*tên kiểu mảng*> = **array[***kiểu chỉ số hàng*, *kiểu chỉ số cột***]**
of <*kiểu phần tử*>;

var <*tên biến mảng*>**:** <*tên kiểu mảng*>;

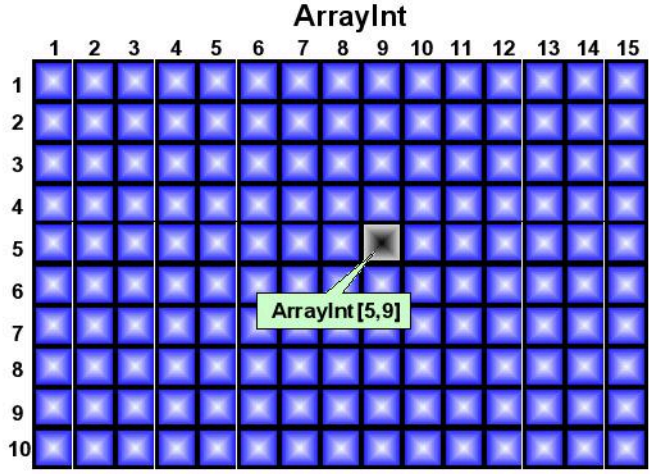
Ví dụ. Các khai báo sau đây là hợp lệ:

```
type
   ArrayReal = array[-100..200,100..200] of real;
   ArrayBoolean = array[-n+1..n+1,n..2*n] of boolean;
var
   ArrayInt: array[1..10,1..15] of integer;
   ArrayLong:array[0..3*(n+1),0..n] of longint;
```

trong đó *n* là hằng nguyên.

Tham chiếu tới phần tử của mảng hai chiều được xác định bởi tên mảng cùng với hai chỉ số được phân cách bởi dấu phẩy và viết trong cặp ngoặc [và].

Ví dụ. Tham chiếu tới phần tử ở hàng thứ 5, cột thứ 9 của biến mảng *ArrayInt* khai báo trong ví dụ trên được viết: *ArrayInt* [5, 9].



Hình 13. *Minh hoạ mảng hai chiều*

b) *Một số ví dụ*

Ví dụ 1. Chương trình sau tính và đưa ra màn hình bảng nhân.

```
program Bang_nhan;
uses crt;
var
   B: array[1..9,1..10] of integer;
   {B: bien mang hai chieu luu bang nhan}
   i, j: integer;
begin
   clrscr;
   for i:=1 to 9 do
      for j:= 1 to 10 do
         B[i,j]:= i*j;
   for i:=1 to 9 do
```

```
        begin
            for j:=1 to 10 do write(B[i,j]:4);
            writeln;
        end;
    readln
end.
```

Ví dụ 2. Chương trình sau nhập vào từ bàn phím các phần tử của mảng hai chiều *B* gồm 5 hàng, 7 cột với các phần tử là các số nguyên và một số nguyên *k*. Sau đó, đưa ra màn hình các phần tử của mảng có giá trị nhỏ hơn *k*.

```
program MangHaiChieu;
uses crt;
var b: array[1..5, 1..7] of integer;
   d, i, j, k: integer;
begin
   clrscr;
   writeln('Nhap cac phan tu cua mang theo dong: ');
   for i:= 1 to 5 do
     begin
       for j:= 1 to 7 do read(b[i,j]);
       writeln;
     end;
   write('Nhap vao gia tri k = '); readln(k);
   d:= 0;
   writeln('DS cac phan tu mang nho hon ',k,':');
   for i:= 1 to 5 do
      for j:= 1 to 7 do
         if b[i,j] < k then begin
                                write(b[i,j], ' ');
                                d:= d+1;
                            end;
   if d = 0 then writeln('Khong co phan tu nao nho hon ',k);
   readln
end.
```

Chú ý: – Các biến mảng thường gồm số lượng lớn các phần tử nên cần lưu ý phạm vi sử dụng chúng để khai báo kích thước và kiểu dữ liệu sao cho tiết kiệm bộ nhớ.

– Ngoài hai kiểu mảng một chiều và hai chiều, còn có kiểu mảng nhiều chiều.

Bài tập và thực hành 3

1. Mục đích, yêu cầu

- *Nâng cao kĩ năng sử dụng một số câu lệnh và một số kiểu dữ liệu thông qua việc tìm hiểu, chạy thử các chương trình có sẵn;*
- *Biết giải một số bài toán tính toán, tìm kiếm đơn giản trên máy tính.*

2. Nội dung

Bài 1. Tạo mảng A gồm n ($n \leq 100$) số nguyên, mỗi số có giá trị tuyệt đối không vượt quá 300. Tính tổng các phần tử của mảng là bội số của một số nguyên dương k cho trước.

a) Hãy tìm hiểu và chạy thử chương trình sau đây:

```
program Sum1;
uses crt;
const nmax=100;
type MyArray=array[1..nmax] of integer;
var A: MyArray;
   s,n,i,k: integer;
begin
   clrscr; randomize;
   write('Nhap n =');
   readln(n);   {Tao ngau nhien mang gom n so nguyen}
   for i:= 1 to n do A[i]:= random(301)- random(301);
   for i:= 1 to n do write(A[i]:5); {in ra mang vua tao}
   writeln;
   write('Nhap k = ');
   readln(k);
   s:=0;
   for i:=1 to n do
      if A[i] mod k = 0 then s:=s+A[i];
   writeln('Tong can tinh la: ',s);
   readln
end.
```

Chú ý: Hàm chuẩn *random*(*n*) cho giá trị là số nguyên ngẫu nhiên trong đoạn từ 0 đến *n* – 1, còn thủ tục *randomize* khởi tạo cơ chế sinh số ngẫu nhiên.

b) Hãy đưa các câu lệnh sau đây vào những vị trí cần thiết nhằm sửa đổi chương trình trong câu *a)* để có được chương trình đưa ra số các số dương và số các số âm trong mảng.

```
posi, neg: integer;
posi:= 0; neg:= 0;
if A[i]>0 then posi:= posi+1
   else if A[i]<0 then neg:= neg+1;
writeln(posi:4,neg:4);
```

Bài 2. Viết chương trình tìm phần tử có giá trị lớn nhất của mảng và đưa ra màn hình chỉ số và giá trị của phần tử tìm được. Nếu có nhiều phần tử có cùng giá trị lớn nhất thì đưa ra phần tử có chỉ số nhỏ nhất.

a) Hãy tìm hiểu chương trình sau đây:

```
program MaxElement;
const Nmax= 100;
type MyArray = array[1..Nmax] of integer;
var A: MyArray;
   n,i,j: integer;
begin
    write('Nhap so luong phan tu cua day so, N = ');
    readln(N);
    for i:=1 to N do
       begin
         write('Phan tu thu ',i,' = ');
         readln(A[i]);
       end;
   j:= 1;
   for i:= 2 to n do if A[i] > A[j] then j:= i;
   write('Chi so: ',j, ' Gia tri: ',A[j]:4);
   readln
end.
```

b) Chỉnh sửa chương trình trên để đưa ra chỉ số của các phần tử có cùng giá trị lớn nhất.

Bài tập và thực hành 4

1. Mục đích, yêu cầu

- *Biết nhận xét, phân tích, đề xuất thuật toán giải bài toán sao cho chương trình chạy nhanh hơn;*
- *Làm quen với dữ liệu có cấu trúc và bài toán sắp xếp.*

2. Nội dung

Bài 1

a) Hãy tìm hiểu và chạy thử chương trình thực hiện thuật toán sắp xếp dãy số nguyên bằng thuật toán tráo đổi với các giá trị khác nhau của *n* dưới đây. Qua đó, nhận xét về thời gian chạy của chương trình.

```
(* Chuong trinh giai bai toan sap xep day so *)
uses crt;
const  Nmax = 250;
type   ArrInt = array[1..Nmax] of integer;
var    n,i, j, t: integer;
       A: ArrInt;
begin
   clrscr;
   randomize;
   write('Nhap n = ');
   readln(n);
   {Tao ngau nhien mang gom n so nguyen}
   for i:= 1 to n do A[i]:= random(300)-random(300);
   for i:= 1 to n do write(A[i]:5); {in mang vua tao}
   writeln;
   for j:=N downto 2 do
          for i:=1 to j-1 do
```

```
            if A[i]> A[i+1] then
               begin (* Trao doi A[i] va A[i+1] *)
                  t:= A[i];
                  A[i]:= A[i+1];
                  A[i+1]:= t;
               end;
   writeln('Day so duoc sap xep: ');
   for i:=1 to n do
      write(A[i]: 7);
   writeln;
   readln
end.
```

b) Khai báo thêm biến nguyên *Dem* và bổ sung vào chương trình những câu lệnh cần thiết để biến *Dem* tính số lần thực hiện tráo đổi trong thuật toán. Đưa kết quả tìm được ra màn hình.

Bài 2

Hãy đọc và tìm hiểu những phân tích để viết chương trình giải bài toán:

Cho mảng A gồm n phần tử. Hãy viết chương trình tạo mảng B[1..n], trong đó B[i] là tổng của i phần tử đầu tiên của A.

Thoạt đầu có thể viết chương trình sau để giải bài toán này:

```
program SubSum1;
const nmax=100;
type MyArray=array[1..nmax] of integer;
var A, B: MyArray;
   n,i,j: integer;
begin
   randomize;
   write('Nhap n =');
   readln(n);
   {Tao ngau nhien mang gom n so nguyen}
   for i:= 1 to n do A[i]:= random(300) - random(300);
```

```
    for i:= 1 to n do write(A[i]:5);
    writeln;
    {Bat dau tao B}
    for i:= 1 to n do
       begin
          B[i]:= 0;
          for j:= 1 to i do B[i]:= B[i]+A[j];
       end;
    {Ket thuc tao B}
    for i:= 1 to n do write (B[i]:6);
    readln
end.
```

Để ý rằng ta có các hệ thức sau:

```
B[1]= A[1]
B[i]= B[i-1]+A[i], 1<i≤n.
```

Do đó, ta thay đoạn chương trình từ chú thích {*Bat dau tao B*} đến {*Ket thuc tao B*} bởi hai lệnh sau:

```
B[1]:= A[1];
for i:= 2 to n do B[i]:= B[i-1]+A[i];
```

Với hai lệnh này, máy chỉ phải thực hiện $n - 1$ phép cộng, trong khi với đoạn chương trình trên máy phải thực hiện $\frac{n(n+1)}{2}$ phép cộng.

Nhờ việc phân tích như trên ta có thể giảm bớt đáng kể số phép toán cần thực hiện.

Tuy tốc độ tính toán của máy tính nhanh nhưng có giới hạn. Do đó, trong khi viết chương trình, ta nên tìm cách viết sao cho chương trình thực hiện càng ít phép toán càng tốt.

§12. KIỂU XÂU

Dữ liệu trong các bài toán không chỉ thuộc kiểu số mà cả kiểu phi số - dạng kí tự. Dữ liệu kiểu xâu là dãy các kí tự.

Ví dụ. Các xâu kí tự đơn giản:

'Bach khoa' 'KI SU' '2007 la nam Dinh Hoi'

Xâu là dãy các kí tự trong bộ mã ASCII, mỗi kí tự được gọi là một phần tử của xâu. Số lượng kí tự trong một xâu được gọi là độ dài của xâu. Xâu có độ dài bằng 0 gọi là xâu rỗng.

Các ngôn ngữ lập trình đều có quy tắc, cách thức cho phép xác định:

- Tên kiểu xâu;
- Cách khai báo biến kiểu xâu;
- Số lượng kí tự của xâu;
- Các phép toán thao tác với xâu;
- Cách tham chiếu tới phần tử của xâu.

Có thể xem xâu là mảng một chiều mà mỗi phần tử là một kí tự. Các kí tự của xâu được đánh số thứ tự, thường bắt đầu từ 1.

Tương tự mảng, tham chiếu tới phần tử của xâu được xác định bởi tên biến xâu và chỉ số đặt trong cặp ngoặc [và].

Ví dụ, giả sử có biến *Hoten* lưu trữ giá trị hằng xâu '*Nguyen Le Huyen*' thì *Hoten*[6] cho ta kí tự '*n*' là kí tự thứ sáu của biến xâu *Hoten*.

Dưới đây trình bày cách khai báo dữ liệu kiểu xâu, các thao tác xử lí xâu và một số ví dụ sử dụng kiểu xâu trong Pascal.

1. Khai báo

Để khai báo dữ liệu kiểu xâu ta sử dụng tên dành riêng ***string***, tiếp theo là độ dài lớn nhất của xâu (không vượt quá 255) được ghi trong cặp ngoặc [và].

Biến kiểu xâu có thể khai báo như sau:

var <*tên biến*>**: string[***độ dài lớn nhất của xâu***];**

Ví dụ

```
var Hoten: string[26];
```

Trong mô tả xâu có thể bỏ qua phần khai báo độ dài, chẳng hạn:

```
var Chugiai: string;
```

Khi đó độ dài lớn nhất của xâu sẽ nhận giá trị ngầm định là 255.

2. Các thao tác xử lí xâu

a) Phép ghép xâu, kí hiệu là dấu cộng (+), được sử dụng để ghép nhiều xâu thành một. Có thể thực hiện phép ghép xâu đối với các hằng và biến xâu.

Ví dụ

Phép ghép xâu:

```
'Ha' + ' Noi' + ' - ' + 'Viet Nam'
```

cho xâu kết quả là 'Ha Noi - Viet Nam'.

b) Các phép so sánh bằng (=), khác (<>), nhỏ hơn (<), lớn hơn (>), nhỏ hơn hoặc bằng (<=), lớn hơn hoặc bằng (>=) có thứ tự ưu tiên thực hiện thấp hơn phép ghép xâu và thực hiện việc so sánh hai xâu theo các quy tắc sau:

- Xâu *A* là lớn hơn xâu *B* nếu như kí tự đầu tiên khác nhau giữa chúng kể từ trái sang trong xâu *A* có mã ASCII lớn hơn.
- Nếu *A* và *B* là các xâu có độ dài khác nhau và *A* là đoạn đầu của *B* thì *A* là nhỏ hơn *B*.

Ví dụ

```
'May tinh' < 'May tinh cua toi'
```

- Hai xâu được coi là bằng nhau nếu như chúng giống nhau hoàn toàn.

Ví dụ

```
'TIN HOC' = 'TIN HOC'
```

Để xử lí các xâu, có thể sử dụng các thủ tục và hàm chuẩn dưới đây:

c) Thủ tục *delete*(*st*, *vt*, *n*) thực hiện việc xoá *n* kí tự của biến xâu *st* bắt đầu từ vị trí *vt*.

Ví dụ

Giá trị **st**	Thao tác	Kết quả
'abcdef'	delete(st, 5, 2);	'abcd'
'Song Hong'	delete(st, 1, 5);	'Hong'

d) Thủ tục *insert*(*s*1, *s*2, *vt*) chèn xâu *s*1 vào xâu *s*2, bắt đầu ở vị trí *vt*.

Ví dụ

Giá trị **s1**	Giá trị **s2**	Thao tác	Kết quả
' PC '	'IBM486'	insert(s1,s2,4);	'IBM PC 486'
'1'	'Hinh .2'	insert(s1,s2,6);	'Hinh 1.2'

e) Hàm *copy*(*S*, *vt*, *N*) tạo xâu gồm *N* kí tự liên tiếp bắt đầu từ vị trí *vt* của xâu *S*.

Ví dụ

Giá trị **s**	Biểu thức	Kết quả
'Bai hoc thu 9'	copy(s,9,5);	'thu 9'

f) Hàm *length*(*s*) cho giá trị là độ dài xâu *s*.

Ví dụ

Giá trị **s**	Biểu thức	Kết quả
'500 ki tu'	length(s)	9

g) Hàm *pos*(*s*1, *s*2) cho vị trí xuất hiện đầu tiên của xâu *s*1 trong xâu *s*2.

Ví dụ

Giá trị **s2**	Biểu thức	Kết quả
'abcdef'	pos('cd',s2)	3
'abcdef'	pos('k',s2)	0

h) Hàm *upcase*(*ch*) cho chữ cái in hoa ứng với chữ cái trong *ch*.

Ví dụ

Giá trị ch	Biểu thức	Kết quả
'd'	upcase(ch)	'D'
'E'	upcase(ch)	'E'

3. Một số ví dụ

Ví dụ 1

Chương trình dưới đây nhập họ tên của hai người vào hai biến xâu và đưa ra màn hình xâu dài hơn, nếu bằng nhau thì đưa ra xâu nhập sau.

```
var
   a,b:string;
begin
   write('Nhap ho ten thu nhat: '); readln(a);
   write('Nhap ho ten thu hai: '); readln(b);
   if length(a)>length(b) then write(a) else write(b);
   readln
end.
```

Ví dụ 2

Chương trình dưới đây nhập hai xâu từ bàn phím và kiểm tra kí tự đầu tiên của xâu thứ nhất có trùng với kí tự cuối cùng của xâu thứ hai không.

```
var x: byte;
   a,b: string;
begin
   write('Nhap xau thu nhat: ');
   readln(a);
   write('Nhap xau thu hai: ');
   readln(b);
   x:=length(b);
   {xac dinh do dai xau b de biet vi tri cua ki tu cuoi cung}
   if a[1]=b[x] then write('Trung nhau')
                else write('Khac nhau');
   readln
end.
```

Ví dụ 3

Chương trình sau nhập một xâu vào từ bàn phím và đưa ra màn hình xâu đó nhưng được viết theo thứ tự ngược lại.

```
var i,k: byte;
   a: string;
begin
   write('Nhap xau:');
   readln(a);
   k:= length(a);{xac dinh do dai xau}
   for i:= k downto 1 do write(a[i]);
   readln
end.
```

Ví dụ 4

Chương trình sau nhập một xâu vào từ bàn phím và đưa ra màn hình xâu thu được từ nó sau khi loại bỏ các dấu cách nếu có.

```
var i,k: byte;
   a, b: string;
begin
   write('Nhap xau:');
   readln(a);
   k:= length(a);
   b:= ''; (* Khoi tao xau rong *)
   for i:= 1 to k do
      if a[i]<>' ' then b:=b+a[i];
   writeln('Ket qua: ',b);
   readln
end.
```

Ví dụ 5

Chương trình sau nhập vào từ bàn phím xâu kí tự *s*1, tạo xâu *s*2 gồm tất cả các chữ số có trong *s*1 (giữ nguyên thứ tự xuất hiện của chúng) và đưa kết quả ra màn hình.

```
program XuLiXau;
var s1, s2: string;
   i: byte;
begin
   write('Nhap vao xau s1: ');
   readln(s1);
   s2:= ''; {Khoi tao xau s2 rong}
   for i:= 1 to length(s1) do
      if ('0'<=s1[i]) and (s1[i]<= '9') then s2:= s2+s1[i];
   writeln('Ket qua: ', s2);
   readln
end.
```

Bài tập và thực hành 5

1. Mục đích, yêu cầu

Làm quen với việc tìm kiếm, thay thế và biến đổi xâu.

2. Nội dung

Bài 1. Nhập vào từ bàn phím một xâu. Kiểm tra xâu đó có phải là xâu đối xứng hay không. Xâu đối xứng có tính chất: đọc nó từ phải sang trái cũng thu được kết quả giống như đọc từ trái sang phải (còn được gọi là xâu palindrome).

a) Hãy chạy thử chương trình sau:

```
var i,x: byte;
    a,p: string;
begin
    write('Nhap vao xau: ');
    readln(a);
    x:=length(a);    {xac dinh do dai cua xau }
    p:='';           {khoi tao xau rong }
    for i:=x downto 1 do
        p:=p+a[i];   {tao xau dao nguoc }
    if a=p then
        write('Xau la palindrome')
    else
        write('Xau khong la palindrome');
    readln
end.
```

b) Hãy viết lại chương trình trên, trong đó không dùng biến xâu *p*.

Bài 2. Viết chương trình nhập từ bàn phím một xâu kí tự *S* và thông báo ra màn hình số lần xuất hiện của mỗi chữ cái tiếng Anh trong *S* (không phân biệt chữ hoa hay chữ thường).

Bài 3. Nhập vào từ bàn phím một xâu. Thay thế tất cả các cụm kí tự '*anh*' bằng cụm kí tự '*em*'.

§13. KIỂU BẢN GHI

Dữ liệu kiểu *bản ghi* (record) dùng để mô tả các đối tượng có cùng một số thuộc tính mà các thuộc tính có thể có các kiểu dữ liệu khác nhau.

Ví dụ, bảng kết quả thi gồm thông tin về các thí sinh như họ và tên, ngày sinh, giới tính, điểm các môn thi,... mà những thông tin này thuộc các kiểu dữ liệu khác nhau.

Bảng kết quả thi

Họ và tên	Ngày sinh	Giới tính	Điểm Tin	Điểm Toán	Điểm Lí	Điểm Hoá	Điểm Văn	Điểm Sử	Điểm Địa
Nguyễn Thị Minh Huệ	12/12/1990	Nữ	9	10	7	8	8	7	8
Dương Trúc Lâm	2/1/1990	Nam	9	10	8	8	9	6	7
Đào Văn Bình	5/12/1990	Nam	8	8	9	8	7	7	6
...	...	...	...	...	...	...	...	...	...

Một ví dụ khác, khi xem xét doanh số của một cửa hàng, ta quan tâm đến tập hoá đơn bán hàng, mỗi hoá đơn đều có các thuộc tính như tên hàng, đơn giá, chủng loại, số lượng bán, giá thành, người bán, người mua, ngày bán,...

Để mô tả các đối tượng như vậy, ngôn ngữ lập trình cho phép xác định kiểu dữ liệu bản ghi (trong C++ gọi là kiểu cấu trúc (struct)). Mỗi đối tượng được mô tả bằng một bản ghi. Mỗi thuộc tính của đối tượng tương ứng với một *trường* của bản ghi. *Các trường khác nhau có thể có các kiểu dữ liệu khác nhau*.

Ngôn ngữ lập trình đưa ra quy tắc, cách thức xác định:

- Tên kiểu bản ghi;
- Tên các thuộc tính (trường);
- Kiểu dữ liệu của mỗi trường;
- Cách khai báo biến;
- Cách tham chiếu đến trường.

Dưới đây giới thiệu cách khai báo kiểu, biến, tham chiếu đến trường và phép gán giá trị bản ghi trong Pascal.

1. Khai báo

Các thông tin cần khai báo bao gồm tên kiểu bản ghi, tên các thuộc tính, kiểu dữ liệu của mỗi thuộc tính.

Do dữ liệu kiểu bản ghi thường dùng để mô tả nhiều đối tượng nên ta thường định nghĩa một kiểu bản ghi và sau đó dùng nó để khai báo các biến liên quan.

Kiểu bản ghi thường được định nghĩa như sau:

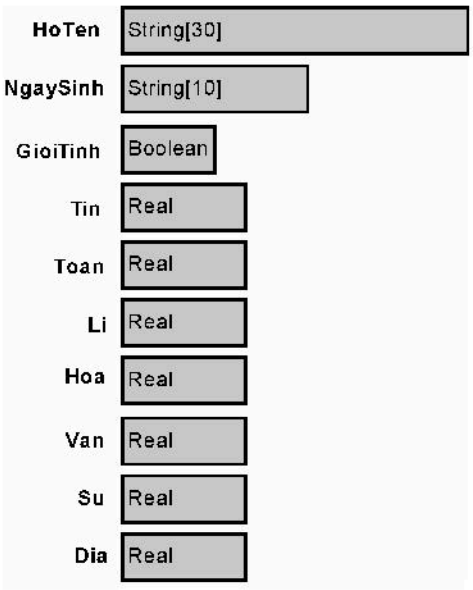
```
type <tên kiểu bản ghi> = record
                            <tên trường 1>:<kiểu trường 1>;
                            .....
                            <tên trường k>:<kiểu trường k>;
                          end;
```

Sau khi có kiểu bản ghi, biến kiểu bản ghi có thể được khai báo như sau:

```
var
    <tên biến bản ghi>: <tên kiểu bản ghi>;
```

Ví dụ

Để xử lí bảng kết quả thi nêu trên ta có thể khai báo *Lop* là biến mảng một chiều, mỗi phần tử mảng là một bản ghi *HocSinh* (dữ liệu về một học sinh). Mỗi bản ghi *HocSinh* gồm các thông tin: *HoTen*, *NgaySinh*, *GioiTinh* và điểm 7 môn thi: *Tin*, *Toan*, *Li*, *Hoa*, *Van*, *Su*, *Dia*.



Hình 14. *Bản ghi HocSinh*

Trong chương trình xử lí kết quả thi có thể sử dụng khai báo sau đây:

```
const Max =60; {gia thiet si so lop cao nhat la 60}
type
   HocSinh = record
               HoTen: string[30];
               NgaySinh: string[10];
               GioiTinh: boolean;
               Tin, Toan, Li, Hoa, Van, Su, Dia: Real;
             end;
var
   A, B: HocSinh;
   Lop: array[1..Max] of HocSinh;
```

Nếu *A* là biến kiểu bản ghi và *X* là tên một trường của *A*, thì để tham chiếu đến trường *X*, ta viết:

```
A.X
```

Để tham chiếu đến điểm tin học của một học sinh trong ví dụ trên ta viết:

```
A.Tin
```

2. Gán giá trị

Có hai cách để gán giá trị cho biến bản ghi:

- *Dùng lệnh gán trực tiếp:* Nếu *A* và *B* là hai biến bản ghi cùng kiểu, thì ta có thể gán giá trị của *B* cho *A* bằng câu lệnh:

```
A:= B;
```

- *Gán giá trị cho từng trường:* Có thể thực hiện bằng lệnh gán hoặc nhập từ bàn phím.

Ví dụ, một lớp gồm N ($N \leq 60$) học sinh. Cần quản lí học sinh với các thuộc tính như họ và tên, ngày sinh, địa chỉ, điểm toán, điểm văn, xếp loại. Giả sử việc xếp loại được xác định như sau:

- Nếu tổng điểm toán và văn lớn hơn hoặc bằng 18 thì xếp loại *A*.
- Nếu tổng điểm toán và văn lớn hơn hoặc bằng 14 và nhỏ hơn 18 thì xếp loại *B*.

- Nếu tổng điểm toán và văn lớn hơn hoặc bằng 10 và nhỏ hơn 14 thì xếp loại *C*.
- Nếu tổng điểm toán và văn nhỏ hơn 10 thì xếp loại *D*.

Chú ý rằng, trong các thuộc tính cần quản lí, chỉ có năm thuộc tính đầu là độc lập, còn thuộc tính xếp loại được xác định dựa vào các điểm toán và văn. Để lưu trữ thông tin về học sinh, ta dùng kiểu bản ghi với sáu trường tương ứng với sáu thuộc tính cần quản lí.

Dưới đây là chương trình nhập vào từ bàn phím thông tin của từng học sinh trong lớp, thực hiện xếp loại và đưa ra màn hình kết quả xếp loại học sinh:

```
program Xep_loai;
uses crt;
const Max = 60;
type   HocSinh = record
                    HoTen:string[30];
                    NgaySinh:string[10];
                    DiaChi:string[50];
                    Toan,Van:real;
                    XepLoai:char;
                 end;
var
   Lop: array[1..Max] of HocSinh;
   N,i: Byte;
begin
   clrscr;
   write('So luong hoc sinh trong lop N = ');readln(N);
   for i:= 1 to N do
      begin
         writeln('Nhap so lieu ve hoc sinh thu ',i,': ');
         write('Ho va ten: ');readln(Lop[i].HoTen);
         write('Ngay sinh: ');readln(Lop[i].NgaySinh);
         write('Dia chi: ');readln(Lop[i].DiaChi);
         write('Diem Toan: ');readln(Lop[i].Toan);
         write('Diem Van: ');readln(Lop[i].Van);
         if Lop[i].Toan+Lop[i].Van>=18
            then Lop[i].XepLoai:='A';
         if (Lop[i].Toan+Lop[i].Van>=14)and
            (Lop[i].Toan+Lop[i].Van<18)
            then Lop[i].XepLoai:='B';
```

```
            if (Lop[i].Toan+Lop[i].Van>=10)and
               (Lop[i].Toan+Lop[i].Van<14)
               then Lop[i].XepLoai:='C';
            if (Lop[i].Toan+Lop[i].Van<10)
               then Lop[i].XepLoai:='D';
         end;
      clrscr;
      writeln('Danh sach xep loai hoc sinh trong lop:');
      for i:= 1 to N do
        writeln(Lop[i].HoTen:30,' - Xep loai: ',Lop[i].XepLoai);
      readln
end.
```

TÓM TẮT

- Kiểu dữ liệu có cấu trúc được xây dựng từ những kiểu dữ liệu đã có theo quy tắc, khuôn dạng do ngôn ngữ lập trình cung cấp.
- Mảng một chiều
  - Mảng một chiều là dãy hữu hạn các phần tử cùng kiểu.
  - Khai báo: tên mảng, kiểu chỉ số, kiểu phần tử.
  - Tham chiếu phần tử mảng: *tên biến mảng*[*chỉ số phần tử*]
- Mảng hai chiều
  - Mảng hai chiều là bảng các phần tử cùng kiểu.
  - Khai báo: tên mảng, kiểu chỉ số hàng, kiểu chỉ số cột, kiểu phần tử.
  - Tham chiếu phần tử mảng: *tên biến mảng*[*chỉ số hàng, chỉ số cột*]
- Kiểu dữ liệu xâu
  - Xâu là dãy các kí tự trong bộ mã ASCII.
  - Các thao tác xử lí thường sử dụng:
    - Phép ghép xâu;
    - Phép so sánh;
    - Các thủ tục và hàm chuẩn xử lí xâu.
- Kiểu bản ghi
  - Khai báo: tên bản ghi, tên và kiểu các trường.
  - Tham chiếu trường của bản ghi: *tên biến bản ghi.tên trường*

CÂU HỎI VÀ BÀI TẬP

1. Tại sao mảng là kiểu dữ liệu có cấu trúc?
2. Tại sao phải khai báo kích thước của mảng?
3. Các phần tử của mảng có thể có những kiểu gì?
4. Tham chiếu đến phần tử của mảng bằng cách nào?
5. Viết chương trình nhập từ bàn phím số nguyên dương N ($N \le 100$) và dãy A gồm N số nguyên $A_1, A_2,..., A_N$ có giá trị tuyệt đối không lớn hơn 1000. Hãy cho biết dãy A có phải là một cấp số cộng hay không và thông báo kết quả ra màn hình.
6. Viết chương trình nhập từ bàn phím số nguyên dương N ($N \le 100$) và dãy A gồm N số nguyên $A_1, A_2,..., A_N$ có trị tuyệt đối không lớn hơn 1000. Hãy đưa ra những thông tin sau:

   a) Số lượng số chẵn và số lẻ trong dãy;

   b) Số lượng số nguyên tố trong dãy.
7. Dãy F là dãy *Phi-bô-na-xi* nếu:

   $F_0 = 0$; $F_1 = 1$; $F_N = F_{N-1} + F_{N-2}$ với $N \ge 2$.

   Viết chương trình nhập từ bàn phím số nguyên dương N và đưa ra màn hình số hạng thứ N của dãy *Phi-bô-na-xi*. Chương trình của bạn thực hiện được với giá trị lớn nhất của N là bao nhiêu?
8. Chương trình sau đây thực hiện những gì?

```
program BT8;
const NMax = 50;
type Mass = array[1..NMax,0..NMax-1] of real;
var A: Mass;
    i, j, N: byte; C: real;
begin
   write('Nhap N =  '); readln(N);
   for i:= 1 to N do
      for j:= 0 to N-1 do
         begin
            write('A[',i,',',j,']= '); readln(A[i,j]);
         end;
    for i:= 1 to N do
      for j:= 0 to N-1 do
         begin
            C:= A[i,j];
            A[i,j]:= A[N-i+1,j];
            A[N-i+1,j]:= C;
         end;
```

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

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

Chương trình sau đây giải bài toán trên:

```
program Diag;
var
    N, i, j, Max,Ind,Vsp: integer;
    A: array[1..15, 1..15] of integer;
begin
    write('Nhap N nho hon 15: '); readln(N);
    for i:= 1 to N do
       for j:= 1 to N do
          begin
            write('A[', i, ',', j, ']= '); readln(A[i,j]);
          end;
    for i:= 1 to N do
       begin
          Max:= A[i,1]; Ind:= 1;
          for j:= 2 to N do
             if A[i,j] > Max then
                begin
                   Max:= A[i,j]; Ind:= j
                end;
          Vsp:= A[i,i]; A[i,i]:= Max; A[i,Ind]:= Vsp;
       end;
    for i:= 1 to N do
       begin
          writeln;
          for j:= 1 to N do write(A[i,j]: 3);
       end;
    writeln
end.
```

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

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

11. Hãy bổ sung thêm vào chương trình *Xep_loai* (ở §13) những lệnh cần thiết để chương trình đưa ra danh sách học sinh xếp loại *A*.

Chương V

TỆP VÀ THAO TÁC VỚI TỆP

§14. KIỂU DỮ LIỆU TỆP

1. Vai trò của kiểu tệp

Tất cả các dữ liệu thuộc các kiểu dữ liệu đã xét đều được lưu trữ ở bộ nhớ trong (RAM) và do đó dữ liệu sẽ bị mất khi tắt máy. Với một số bài toán có khối lượng dữ liệu lớn, có yêu cầu lưu trữ để xử lí nhiều lần, cần có kiểu dữ liệu tệp (file).

Kiểu dữ liệu tệp có những đặc điểm sau:

- Dữ liệu kiểu tệp được lưu trữ lâu dài ở bộ nhớ ngoài (đĩa từ, CD,…) và không bị mất khi tắt nguồn điện;
- Lượng dữ liệu lưu trữ trên tệp có thể rất lớn và chỉ phụ thuộc vào dung lượng đĩa.

2. Phân loại tệp và thao tác với tệp

Xét theo cách tổ chức dữ liệu, có thể phân tệp thành hai loại:

- *Tệp văn bản* là tệp mà dữ liệu được ghi dưới dạng các kí tự theo mã ASCII. Trong tệp văn bản, dãy kí tự kết thúc bởi kí tự xuống dòng hay kí tự kết thúc tệp tạo thành một dòng.

  Các dữ liệu dạng văn bản như sách, tài liệu, bài học, giáo án, các chương trình nguồn viết bằng ngôn ngữ bậc cao,... thường được lưu trữ dưới dạng tệp văn bản.

- *Tệp có cấu trúc* là tệp mà các thành phần của nó được tổ chức theo một cấu trúc nhất định. Tệp nhị phân là một trường hợp riêng của tệp có cấu trúc. Dữ liệu ảnh, âm thanh,... thường được lưu trữ dưới dạng tệp có cấu trúc.

Xét theo cách thức truy cập, có thể phân tệp thành hai loại:

- *Tệp truy cập tuần tự* cho phép truy cập đến một dữ liệu nào đó trong tệp chỉ bằng cách bắt đầu từ đầu tệp và đi qua lần lượt tất cả các dữ liệu trước nó.
- *Tệp truy cập trực tiếp* cho phép tham chiếu đến dữ liệu cần truy cập bằng cách xác định *trực tiếp* vị trí (thường là số hiệu) của dữ liệu đó.

Khác với mảng, số lượng phần tử của tệp không cần xác định trước.

Hai thao tác cơ bản đối với tệp là ghi dữ liệu vào tệp và đọc dữ liệu từ tệp.

Thao tác đọc/ghi với tệp được thực hiện với từng phần tử của tệp.

Để có thể thao tác với kiểu tệp, người lập trình cần tìm hiểu cách thức mà ngôn ngữ lập trình cung cấp để:

- Khai báo biến tệp;
- Mở tệp;
- Đọc/ghi dữ liệu;
- Đóng tệp.

§15. THAO TÁC VỚI TỆP

Trong mục này ta xét cách khai báo, thao tác với tệp văn bản trong Pascal.

1. Khai báo

Để làm việc với kiểu dữ liệu tệp ta phải sử dụng biến tệp.

Khai báo biến tệp văn bản có dạng:

```
var <tên biến tệp>: text;
```

Ví dụ

```
var tep1,tep2: text;
```

Khai báo trên xác định hai biến tệp văn bản *tep*1 và *tep*2.

2. Thao tác với tệp

a) Gắn tên tệp

Mỗi tệp đều có một *tên tệp* để tham chiếu. *Tên tệp* là biến xâu hoặc hằng xâu, ví dụ **'DULIEU.DAT'**.

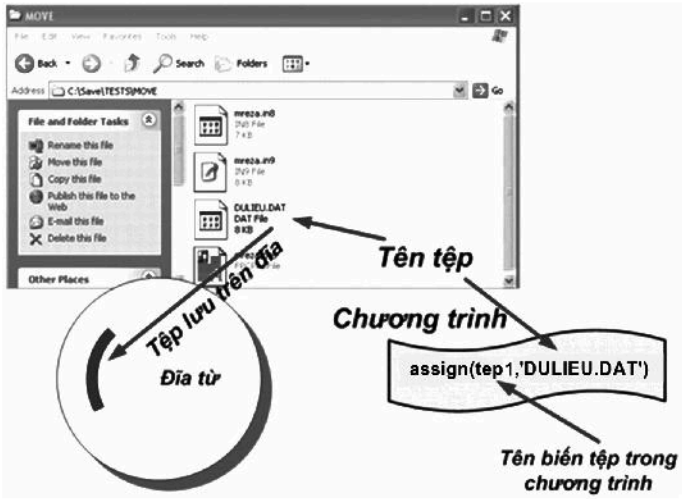
Trong lập trình, ta không thao tác trực tiếp với tệp dữ liệu trên đĩa mà thông qua *biến tệp*. Có thể hình dung biến tệp được ngôn ngữ lập trình sử dụng như đại diện cho tệp.

Do vậy, để thao tác với tệp, trước hết phải gắn tên tệp với đại diện của nó là biến tệp bằng thủ tục:

assign(<*biến tệp*>, <*tên tệp*>);

trong đó, *tên tệp* là biến xâu hoặc hằng xâu.

Sau lệnh này, các thao tác với biến tệp thực chất là thao tác với tệp.



Hình 15. *Quan hệ giữa tệp, biến tệp và tên tệp*

Ví dụ 1

Giả thiết có biến tệp *tep*1 cần gắn với tệp có tên DULIEU.DAT. Để làm điều này ta thực hiện thủ tục sau:

```
assign(tep1,'DULIEU.DAT');
```

Ví dụ 2

Để có thể đọc dữ liệu từ tệp INP.DAT trên thư mục gốc của đĩa C, ta cần gắn tệp đó với một biến tệp, ví dụ là *tep2* bởi thủ tục:

```
assign(tep2,'C:\INP.DAT');
```

b) Mở tệp

Tệp có thể dùng để chứa kết quả ra hoặc dữ liệu vào. Trước khi mở tệp, biến tệp phải được gắn tên tệp bằng thủ tục *assign*.

Câu lệnh dùng thủ tục mở tệp để ghi dữ liệu có dạng:

rewrite(<*biến tệp*>);

Ví dụ

```
assign(tep3,'C:\KQ.DAT');
rewrite(tep3);
```

Khi thực hiện thủ tục *rewrite*(*tep*3), nếu trên thư mục gốc của đĩa C chưa có tệp KQ.DAT, thì tệp sẽ được tạo với nội dung rỗng. Nếu đã có, thì nội dung cũ sẽ bị xoá để chuẩn bị ghi dữ liệu mới.

Trước khi đọc dữ liệu từ tệp đã gắn với một biến tệp, ta mở tệp bằng thủ tục:

```
reset(<biến tệp>);
```

Ví dụ

Để đọc dữ liệu từ tệp DL.INP, ta có thể mở tệp bằng các thủ tục:

```
tentep:= 'DL.INP';
assign(tep1,tentep);
reset(tep1);
```

hoặc

```
assign(tep1, 'DL.INP');
reset(tep1);
```

c) *Đọc/ghi tệp văn bản*

Việc đọc tệp văn bản được thực hiện giống như nhập từ bàn phím. Việc ghi dữ liệu ra tệp văn bản giống như ghi ra màn hình. Dữ liệu trong tệp văn bản được chia thành các dòng.

Câu lệnh dùng thủ tục đọc có dạng:

```
read(<biến tệp>,<danh sách biến>);
```

hoặc

```
readln(<biến tệp >,<danh sách biến>);
```

trong đó, *danh sách biến* là một hoặc nhiều tên biến đơn. Trong trường hợp nhiều biến thì các biến phân cách nhau bởi dấu phẩy.

Câu lệnh dùng thủ tục ghi có dạng:

```
write(<biến tệp>,<danh sách kết quả>);
```

hoặc

```
writeln(<biến tệp>,<danh sách kết quả >);
```

trong đó, *danh sách kết quả* gồm một hoặc nhiều phần tử. Phần tử có thể là biến đơn hoặc biểu thức (số học, quan hệ hoặc lôgic) hoặc hằng xâu. Trường hợp có nhiều phần tử thì các phần tử được phân cách bởi dấu phẩy.

Ví dụ

Giả sử trong chương trình có khai báo:

```
var tepA, tepB: text;
```

và tệp *tepA* được mở để đọc dữ liệu, còn tệp *tepB* dùng để ghi dữ liệu.

Các thủ tục dùng để đọc dữ liệu từ tệp *tepA* có thể như sau:

```
read(tepA,A,B,C);
```

hoặc

```
readln(tepA,X,Y);
```

Các thủ tục dùng để ghi dữ liệu vào tệp *tepB* có thể như sau:

```
write(tepB,' A = ', A,' B = ',B,' C = ',C);
writeln(tepB,' X1 = ',(-B - SQRT(B*B- 4*A*C))/(2*A):8:3);
```

Một số hàm chuẩn thường dùng trong khi đọc/ghi tệp văn bản:

- Hàm **eof**(*<biến tệp>*) trả về giá trị *true* nếu con trỏ tệp đang chỉ tới cuối tệp.
- Hàm **eoln**(*<biến tệp>*) trả về giá trị *true* nếu con trỏ tệp đang chỉ tới cuối dòng.

d) Đóng tệp

Sau khi làm việc xong với tệp cần phải đóng tệp. Việc đóng tệp là đặc biệt quan trọng sau khi ghi dữ liệu, khi đó hệ thống mới thực sự hoàn tất việc ghi dữ liệu ra tệp.

Câu lệnh dùng thủ tục đóng tệp có dạng:

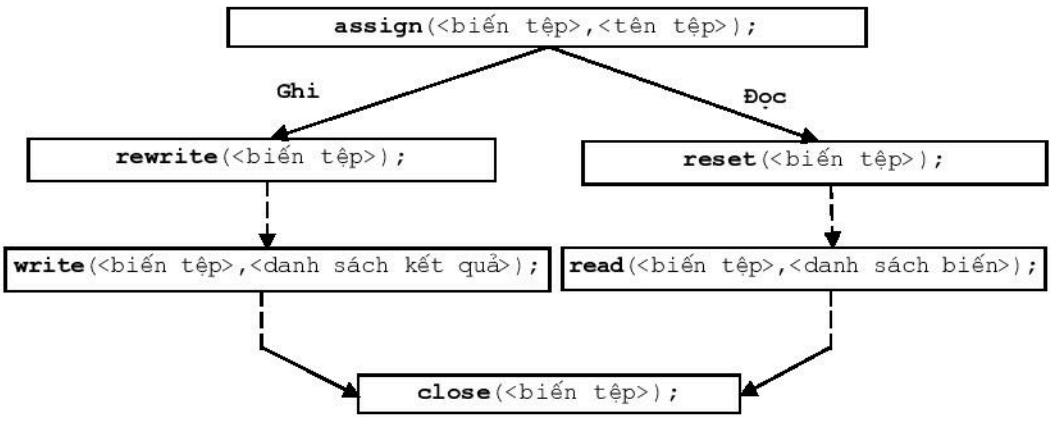
```
close(<biến tệp>);
```

Ví dụ

```
close(tep1);
close(tep3);
```

Sau khi đóng, một tệp vẫn có thể được mở lại. Khi mở lại tệp, nếu vẫn dùng biến tệp cũ thì không cần phải dùng thủ tục *assign* gắn lại tên tệp.

Các thao tác với tệp được mô tả trong hình 16.



Hình 16. *Thao tác với tệp*

§16. VÍ DỤ LÀM VIỆC VỚI TỆP

Ví dụ 1

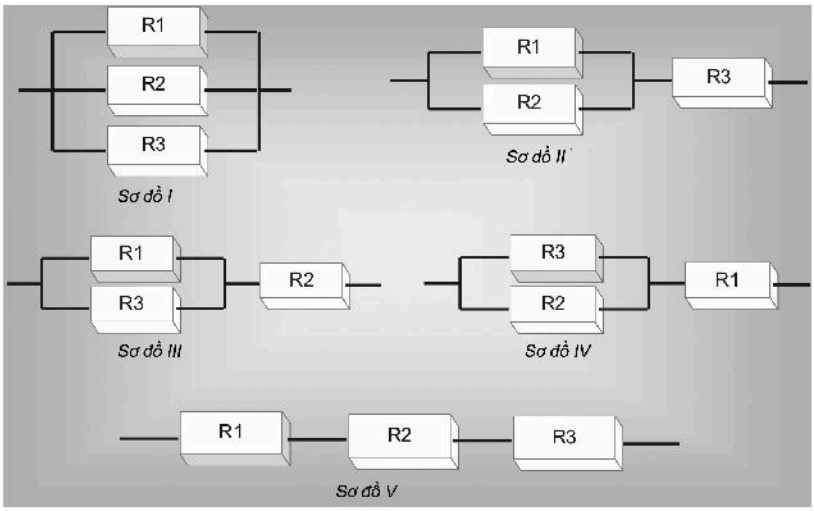
Một trường trung học phổ thông tổ chức cho giáo viên và học sinh của trường đi cắm trại, sinh hoạt ngoài trời ở vườn quốc gia Cúc Phương. Để lên lịch đến thăm khu trại các lớp, thầy hiệu trưởng cần biết khoảng cách từ trại của mình (ở vị trí có toạ độ (0, 0)) đến trại của các giáo viên chủ nhiệm. Mỗi lớp có một khu trại, vị trí trại của mỗi giáo viên chủ nhiệm đều có toạ độ nguyên (x, y) được ghi trong tệp văn bản TRAI.TXT (như vậy, tệp TRAI.TXT chứa liên tiếp các cặp số nguyên, các số cách nhau bởi dấu cách và không kết thúc bằng kí tự xuống dòng).

Chương trình sau sẽ đọc các cặp toạ độ từ tệp TRAI.TXT, tính rồi đưa ra màn hình khoảng cách (với độ chính xác hai chữ số sau dấu chấm thập phân) giữa trại của mỗi giáo viên chủ nhiệm và trại của thầy hiệu trưởng.

```
program Khoang_cach;
var d:real;
    f:text;
    x,y:integer;
begin
   assign(f,'TRAI.TXT');
   reset(f);
   while not eof(f) do
       begin
          read(f,x,y);
          d:=sqrt(x*x+y*y);
          writeln('Khoang cach:', d:10:2)
       end;
   close(f)
end.
```

Ví dụ 2. Tính điện trở tương đương

Cho ba điện trở $R1$, $R2$, $R3$. Sử dụng cả ba điện trở để tạo ra năm mạch điện có điện trở tương đương khác nhau bằng cách mắc theo các sơ đồ nêu ở hình 17.



Hình 17. *Sơ đồ mắc điện trở*

Mỗi cách mắc sẽ cho một điện trở tương đương khác nhau. Ví dụ, nếu mắc theo sơ đồ *I* thì điện trở tương đương sẽ là:

$$R = \frac{R1 * R2 * R3}{R1 * R2 + R1 * R3 + R2 * R3}$$

Nếu mắc theo sơ đồ *V* thì $R = R1 + R2 + R3$.

Cho tệp văn bản RESIST.DAT gồm nhiều dòng, mỗi dòng chứa ba số thực $R1$, $R2$ và $R3$, các số cách nhau một dấu cách, $0 < R1, R2, R3 \le 10^5$.

Chương trình sau đọc dữ liệu từ tệp RESIST.DAT, tính các điện trở tương đương và ghi kết quả ra tệp văn bản RESIST.EQU, mỗi dòng ghi năm điện trở tương đương của ba điện trở ở dòng dữ liệu vào tương ứng.

```
program Dientro;
var a:array[1..5] of real;
    R1,R2,R3:Real;
    i:integer;
    f1,f2:text;
begin
  assign(f1,'RESIST.DAT');
  reset(f1);
  assign(f2,'RESIST.EQU');
  rewrite(f2);
  while not eof(f1) do
```

```
    begin
      readln(f1,R1,R2,R3);
      a[1]:=R1*R2*R3/(R1*R2+R1*R3+R2*R3);
      a[2]:=R1*R2/(R1+R2)+R3;
      a[3]:=R1*R3/(R1+R3)+R2;
      a[4]:=R2*R3/(R2+R3)+R1;
      a[5]:=R1+R2+R3;
      for i:=1 to 5 do write(f2,a[i]:9:3,' ');
      writeln(f2);
    end;
  close(f1); close(f2)
end.
```

TÓM TẮT

- Việc trao đổi dữ liệu với bộ nhớ ngoài được thực hiện thông qua kiểu dữ liệu tệp.
- Để có thể làm việc với tệp cần phải khai báo biến tệp.
- Mỗi ngôn ngữ lập trình đều có các hàm/thủ tục chuẩn để làm việc với tệp.
- Các thao tác với tệp văn bản:
  - Khai báo biến tệp, mở tệp và đóng tệp.
  - Đọc/ghi: tương tự như làm việc với bàn phím và màn hình.

CÂU HỎI VÀ BÀI TẬP

1. Nêu một số trường hợp cần phải dùng tệp.
2. Trong sơ đồ thao tác với tệp, khi cần ghi dữ liệu vào tệp phải dùng những thao tác nào?
3. Tại sao cần phải có câu lệnh mở tệp trước khi đọc/ghi tệp?
4. Tại sao phải dùng câu lệnh đóng tệp sau khi đã kết thúc ghi dữ liệu vào tệp?

Chương VI

CHƯƠNG TRÌNH CON VÀ LẬP TRÌNH CÓ CẤU TRÚC

§17. CHƯƠNG TRÌNH CON VÀ PHÂN LOẠI

1. Khái niệm chương trình con

Các chương trình giải các bài toán phức tạp thường rất dài, có thể gồm hàng trăm, hàng nghìn lệnh. Khi đọc những chương trình dài, rất khó nhận biết được chương trình thực hiện các công việc gì và việc hiệu chỉnh chương trình cũng khó khăn. Vì vậy, vấn đề đặt ra là phải cấu trúc chương trình như thế nào để cho chương trình dễ đọc, dễ hiệu chỉnh, dễ nâng cấp.

Mặt khác, việc giải quyết một bài toán phức tạp thường đòi hỏi và nói chung có thể phân thành các bài toán con.

Xét bài toán tính tổng bốn luỹ thừa:

$$TLuythua = a^n + b^m + c^p + d^q$$

Bài toán trên bao gồm bốn bài toán con tính a^n, b^m, c^p, d^q, có thể giao cho bốn người, mỗi người thực hiện một bài. Giá trị *TLuythua* là tổng kết quả của bốn bài toán con đó. Với những bài toán phức tạp hơn, mỗi bài toán con lại có thể được phân chia thành các bài toán con nhỏ hơn. Quá trình phân rã làm "mịn" dần bài toán như vậy được gọi là cách thiết kế từ trên xuống.

Tương tự, khi lập trình để giải bài toán trên máy tính có thể phân chia chương trình (gọi là chương trình chính) thành các khối (môđun), mỗi khối bao gồm các lệnh giải một bài toán con nào đó. Mỗi khối lệnh sẽ được xây dựng thành một *chương trình con*. Sau đó, chương trình chính sẽ được xây dựng từ các chương trình con này. Chương trình con cũng có thể được xây dựng từ các chương trình con khác.

Cách lập trình như vậy dựa trên phương pháp lập trình có cấu trúc và chương trình được xây dựng gọi là chương trình có cấu trúc.

Chương trình con là một dãy lệnh mô tả một số thao tác nhất định và có thể được thực hiện (được gọi) từ nhiều vị trí trong chương trình.

Ví dụ, chương trình nhập dữ liệu từ bàn phím, tính và đưa ra màn hình giá trị *TLuythua* được mô tả như trên với *a*, *b*, *c*, *d* có kiểu thực và *m*, *n*, *p*, *q* có kiểu nguyên có thể viết bằng Pascal như sau:

```
program tinh_tong;
var TLuythua, Luythua1, Luythua2, Luythua3, Luythua4: real;
   a,b,c,d: real;
   i,n,m,p,q:integer;
begin
   write('Hay nhap du lieu theo thu tu a,b,c,d,m,n,p,q');
   readln(a,b,c,d,m,n,p,q);
   Luythua1:=1.0;
      for i:=1 to n do
         Luythua1:=Luythua1*a;
   Luythua2:=1.0;
      for i:=1 to m do
         Luythua2:=Luythua2*b;
   Luythua3:=1.0;
      for i:=1 to p do
         Luythua3:=Luythua3*c;
   Luythua4:=1.0;
      for i:=1 to q do
         Luythua4:=Luythua4*d;
   TLuythua:= Luythua1+Luythua2+Luythua3+Luythua4;
   writeln('Tong luy thua = ', TLuythua:8:4);
   readln
end.
```

Trong chương trình trên có bốn đoạn lệnh tương tự nhau, việc lặp lại những đoạn lệnh tương tự nhau làm cho chương trình vừa dài vừa khó theo dõi. Để nâng cao hiệu quả lập trình, các ngôn ngữ lập trình bậc cao đều cung cấp khả năng xây dựng chương trình con dạng tổng quát "đại diện" cho nhiều đoạn lệnh tương tự nhau, chẳng hạn tính luỹ thừa *Luythua* = x^k, trong đó *Luythua* và *x* là giá trị kiểu thực còn *k* thuộc kiểu nguyên:

```
var j: integer;
Tich:=1.0;
   for j:=1 to k do
      Tich:=Tich*x;
```

Ta có thể đặt tên cho chương trình con này là *Luythua* và tên các biến chứa dữ liệu vào của nó là *x* và *k*. Khi cần tính luỹ thừa của những giá trị cụ thể ta chỉ

cần viết tên gọi chương trình con và thay thế (x, k) bằng giá trị cụ thể tương ứng. Chẳng hạn để tính a^n, b^m, c^p, d^q ta viết *Luythua*(a, n), *Luythua*(b, m), *Luythua*(c, p), *Luythua*(d, q).

Lợi ích của việc sử dụng chương trình con

- *Tránh được việc phải viết lặp đi lặp lại cùng một dãy lệnh* nào đó tương tự như trong ví dụ tính *TLuythua* ở trên. Ngôn ngữ lập trình cho phép tổ chức dãy lệnh đó thành một chương trình con. Sau đó, mỗi khi chương trình chính cần đến dãy lệnh này chỉ cần gọi thực hiện chương trình con đó.
- *Hỗ trợ việc thực hiện các chương trình lớn:* Khi phải viết chương trình lớn hàng nghìn, hàng vạn lệnh, cần huy động nhiều người tham gia, có thể giao cho mỗi người (hoặc mỗi nhóm) viết một chương trình con, rồi sau đó lắp ghép chúng lại thành chương trình chính. Ví dụ, với các bài toán mà việc tổ chức dữ liệu vào và ra không đơn giản thường người ta chia bài toán thành ba bài toán con như nhập, xử lí và xuất dữ liệu, rồi viết các chương trình con tương ứng.
- *Phục vụ cho quá trình trừu tượng hoá:* Người lập trình có thể sử dụng các kết quả được thực hiện bởi chương trình con mà không phải quan tâm đến việc các chương trình con đó được cài đặt như thế nào. Trừu tượng hoá là tư tưởng chủ đạo để xây dựng chương trình nói chung và chương trình có cấu trúc nói riêng.
- *Mở rộng khả năng ngôn ngữ:* Các ngôn ngữ lập trình thường cung cấp phương thức đóng gói các chương trình con nhằm cung cấp như một câu lệnh mới (tương tự như các lệnh gọi thực hiện các hàm và thủ tục chuẩn) cho người lập trình sử dụng mà không cần biết mã nguồn của nó như thế nào. Hiện nay, ngày càng có nhiều thiết bị kĩ thuật số tiện ích như máy quay phim, máy ảnh, máy ghi âm, các thiết bị âm thanh, màn hình màu độ phân giải cao,... có thể được kết nối với máy tính. Việc thiết kế những chương trình con thực hiện các giao tiếp cơ bản với các thiết bị như vậy là rất cần thiết và giúp mở rộng khả năng ứng dụng của ngôn ngữ.
- *Thuận tiện cho phát triển, nâng cấp chương trình:* Do chương trình được tạo thành từ các chương trình con nên chương trình dễ đọc, dễ hiểu, dễ kiểm tra và hiệu chỉnh. Việc nâng cấp, phát triển chương trình con nào đó, thậm chí bổ sung thêm các chương trình con mới nói chung không gây ảnh hưởng đến các chương trình con khác.

2. Phân loại và cấu trúc của chương trình con

a) Phân loại

Trong nhiều ngôn ngữ lập trình, chương trình con thường gồm hai loại:

- *Hàm* (function) là chương trình con thực hiện một số thao tác nào đó và trả về một giá trị qua tên của nó. Ví dụ hàm toán học hay hàm xử lí xâu:

  `sin(x)` nhận giá trị thực x và trả về giá trị $\sin x$,
  `sqrt(x)` nhận giá trị x và trả về giá trị căn bậc hai của x,
  `length(x)` nhận xâu x và trả về độ dài của xâu x,...

- *Thủ tục* (procedure) là chương trình con thực hiện các thao tác nhất định nhưng không trả về giá trị nào qua tên của nó. Ví dụ các thủ tục vào/ra chuẩn hay thủ tục xử lí xâu:

  `writeln`, `readln`, `delete`, `insert`,...

b) Cấu trúc chương trình con

Chương trình con có cấu trúc tương tự chương trình, nhưng nhất thiết phải có tên và phần đầu dùng để khai báo tên, nếu là hàm phải khai báo kiểu dữ liệu cho giá trị trả về của hàm:

<phần đầu>
[*<phần khai báo>*]
<phần thân>

Phần khai báo

Phần khai báo có thể có khai báo biến cho dữ liệu vào và ra, các hằng và biến dùng trong chương trình con.

Phần thân

Phần thân của chương trình con là dãy câu lệnh thực hiện để từ những dữ liệu vào ta nhận được dữ liệu ra hay kết quả mong muốn.

Tham số hình thức

Các biến được khai báo cho dữ liệu vào/ra được gọi là *tham số hình thức* của chương trình con. Các biến được khai báo để dùng riêng trong chương trình con được gọi là *biến cục bộ*.

Ví dụ, trong chương trình con *Luythua(x, k)* ở phần 1 thì x, k là các tham số hình thức và j là biến cục bộ.

Nói chung, chương trình chính và các chương trình con khác không thể sử dụng được các biến cục bộ của một chương trình con, nhưng mọi chương trình con đều sử dụng được các biến của chương trình chính. Do vậy, các biến của chương trình chính được gọi là *biến toàn cục*. Ví dụ, biến *TLuythua* khai báo trong chương trình chính ở ví dụ trên là biến toàn cục.

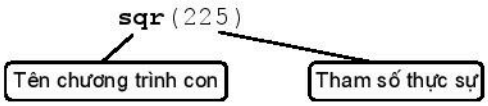
Một chương trình con có thể có hoặc không có tham số hình thức, có thể có hoặc không có biến cục bộ.

c) *Thực hiện chương trình con*

Tham số thực sự

Để thực hiện (gọi) một chương trình con, ta cần phải có lệnh gọi nó tương tự lệnh gọi hàm hay thủ tục chuẩn, bao gồm tên chương trình con với tham số (nếu có) là các hằng và biến chứa dữ liệu vào và ra tương ứng với các tham số hình thức đặt trong cặp ngoặc (và). Các hằng và biến này được gọi là các *tham số thực sự*.

Ví dụ



Khi thực hiện chương trình con, các tham số hình thức dùng để nhập dữ liệu vào sẽ nhận giá trị của tham số thực sự tương ứng, còn các tham số hình thức dùng để lưu trữ dữ liệu ra sẽ trả giá trị đó cho tham số thực sự tương ứng.

Ví dụ, khi thực hiện tính *TLuythua* cần bốn lần gọi chương trình con *Luythua(x, k)* với các tham số (a, n), (b, m), (c, p), (d, q) và các tham số này là tham số thực sự tương ứng với tham số hình thức (x, k).

Sau khi chương trình con kết thúc, lệnh tiếp theo lệnh gọi chương trình con sẽ được thực hiện.

§18. VÍ DỤ VỀ CÁCH VIẾT VÀ SỬ DỤNG CHƯƠNG TRÌNH CON

Các ngôn ngữ lập trình đều có các quy tắc viết và sử dụng chương trình con. Trong mục này, ta xét cách viết và sử dụng chương trình con trong Pascal.

1. Cách viết và sử dụng thủ tục

Xét ví dụ vẽ hình chữ nhật có dạng sau:

```
* * * * * * * *
*               *
* * * * * * * *
```

Ta có thể vẽ hình chữ nhật trên với ba câu lệnh:

```
writeln('* * * * * * * *');
writeln('*             *');
writeln('* * * * * * * *');
```

Như vậy, trong một chương trình, mỗi khi cần vẽ một hình chữ nhật như trên, ta cần phải đưa vào ba câu lệnh này.

Trong chương trình sau, ta đưa ba câu lệnh trên vào một thủ tục có tên là *Ve_Hcn* (vẽ hình chữ nhật). Mỗi khi cần vẽ một hình chữ nhật ta đưa vào một *câu lệnh gọi* thủ tục đó. Chương trình *VD_thutuc1* gọi thủ tục *Ve_Hcn* ba lần để vẽ ba hình chữ nhật.

```
program VD_thutuc1;
procedure Ve_Hcn;   {Bat dau thu tuc}
   begin
      writeln('* * * * * * * *');
      writeln('*             *');
      writeln('* * * * * * * *');
   end;               {Ket thuc thu tuc}
begin
   Ve_Hcn;            {goi thu tuc Ve_Hcn}
```

```
    writeln;writeln;    {de cach 2 dong}
    Ve_Hcn;             {goi thu tuc Ve_Hcn}
    writeln;writeln;
    Ve_Hcn;             {goi thu tuc Ve_Hcn}
end.
```

a) Cấu trúc của thủ tục

Thủ tục có cấu trúc như sau:

procedure <*tên thủ tục*>[(<*danh sách tham số*>)];

[<*phần khai báo*>]

begin

[<*dãy các lệnh*>]

end;

Phần đầu thủ tục gồm tên dành riêng ***procedure***, tiếp theo là tên thủ tục. Danh sách tham số có thể có hoặc không có.

Phần khai báo dùng để xác định các hằng, kiểu, biến và cũng có thể xác định các chương trình con khác được sử dụng trong thủ tục. Thủ tục *Ve_Hcn* ở trên không khai báo hằng, biến hay chương trình con nào.

Dãy câu lệnh được viết giữa cặp tên dành riêng ***begin*** và ***end*** tạo thành thân của thủ tục.

Chú ý: Sau tên dành riêng ***end*** kết thúc chương trình chính là dấu chấm (.) còn sau ***end*** kết thúc thủ tục là dấu chấm phẩy (;).

Các thủ tục, nếu có, phải được khai báo và mô tả trong phần khai báo của chương trình chính, ngay sau phần khai báo các biến.

Khi cần thực hiện, ta phải viết lệnh gọi thủ tục tương tự như các thủ tục chuẩn.

b) Ví dụ về thủ tục

Thủ tục *Ve_Hcn* trong ví dụ trên chỉ vẽ được hình chữ nhật với kích thước cố định là 7×3. Giả sử chương trình cần vẽ nhiều hình chữ nhật với kích thước khác nhau. Để thủ tục *Ve_Hcn* có thể thực hiện được điều đó, cần có hai *tham số* cho dữ liệu vào là chiều dài và chiều rộng. Khi đó phần đầu của thủ tục được viết như sau:

```
procedure Ve_Hcn(chdai,chrong:integer);
```

Khai báo này có nghĩa thủ tục *Ve_Hcn* sẽ được thực hiện để vẽ hình chữ nhật có kích thước tuỳ theo giá trị của các tham số *chdai*, *chrong* và giá trị của các tham số *chdai* và *chrong* là nguyên (*integer*).

Chương trình sau đây mô tả đầy đủ thủ tục *Ve_Hcn* với các tham số *chdai*, *chrong* và sử dụng thủ tục này để vẽ các hình chữ nhật có kích thước khác nhau.

```
program VD_thutuc2;
uses crt;
var a, b, i: integer;
procedure Ve_Hcn(chdai, chrong: integer);
     var i, j: integer;
     begin
        {ve canh tren cua hinh chu nhat}
        for i:= 1 to chdai do write('*');
        writeln;
        for j:=1 to chrong-2 do  {ve 2 canh ben}
          begin
             write('*');
             for i:=1 to chdai-2 do write(' ');
             writeln('*');
          end;
        {ve canh duoi}
        for i:=1 to chdai do write('*');
        writeln;
     end;
begin {bat dau chuong trinh chinh}
   clrscr;
   {ve hinh chu nhat kich thuoc 25×10}
   Ve_Hcn(25,10);
   writeln; writeln; {de cach 2 dong}
   {ve hinh chu nhat kich thuoc 5×10}
   Ve_Hcn(5,10);
   readln;
   clrscr;
   {Ve 4 hinh chu nhat: hinh dau tien co kich thuoc 4×2,
   moi hinh sau co kich thuoc gap doi hinh truoc }
   a:=4; b:=2;
   for i:=1 to 4 do
```

```
        begin
            Ve_Hcn(a,b);
            readln; clrscr;
            a:=a*2; b:= b*2;
        end;
    readln;
end.
```

Trong lệnh gọi thủ tục, các tham số hình thức được thay bằng các tham số thực sự tương ứng là các *giá trị cụ thể* được gọi là các ***tham số giá trị*** (gọi tắt là *tham trị*).

Các tham số *chdai*, *chrong* của thủ tục *Ve_Hcn* là tham trị. Trong lệnh gọi thủ tục *Ve_Hcn*(5, 3) (vẽ hình chữ nhật kích thước 5×3) tham số *chdai* được thay bởi số nguyên 5, tham số *chrong* được thay bởi số nguyên 3.

Còn trong lời gọi thủ tục *Ve_Hcn*(*a*, *b*) vẽ hình chữ nhật kích thước *a*×*b*, tham số *chdai* được thay bởi giá trị hiện thời của biến *a*, tham số *chrong* được thay bởi giá trị hiện thời của biến *b*.

Trong lệnh gọi thủ tục, các tham số hình thức được thay bằng các tham số thực sự tương ứng là tên *các biến* chứa dữ liệu ra được gọi là các ***tham số biến*** (gọi tắt là tham biến).

Để phân biệt tham biến và tham trị, Pascal sử dụng từ khoá ***var*** để khai báo những tham số biến.

Ví dụ, thủ tục *Hoan_doi* trong chương trình sau đổi giá trị của hai biến. Vì cả hai biến đều chứa dữ liệu ra nên cần sử dụng từ khoá ***var*** để khai báo cho cả hai biến.

```
program VD_thambien1;
uses crt;
var a,b: integer;
procedure Hoan_doi(var x,y: integer);
    var TG: integer;
    begin
            TG:= x;
            x:= y;
            y:= TG;
    end;
begin
    clrscr;
    a:=5; b:=10;
```

```
    writeln(a:6,b:6);
    Hoan_doi(a,b);
    writeln(a:6,b:6);
end.
```

Khai báo sau xác định *x* và *y* là hai tham biến kiểu nguyên:

```
    var x,y: integer;
```

Trong lệnh gọi *Hoan_doi*(*a*, *b*) các tham biến *x* và *y* được thay thế bởi các biến nguyên tương ứng *a* và *b*.

Giá trị của các biến này bị thay đổi khi thực hiện các lệnh:

```
    TG:=a; a:=b; b:=TG;
```

Do vậy, sau khi thực hiện thủ tục *Hoan_doi*, biến *a* sẽ nhận giá trị của biến *b* và biến *b* nhận giá trị của biến *a*. Nếu giá trị của *a* là 5 còn giá trị của *b* là 10 thì trên màn hình có hai dòng:

```
5 10
10 5
```

Chương trình sau cho thấy sự khác nhau khi chương trình con dùng tham số giá trị và tham số biến (có và không khai báo với từ khoá *var*):

```
program VD_thambien2;
uses crt;
var a, b: integer;
procedure Hoan_doi(x: integer; var y: integer);
    var TG: integer;
    begin
        TG:=x;
        x:=y;
        y:=TG;
    end;
begin
    clrscr;
    a:=5; b:=10;
    writeln(a:6, b:6);
    Hoan_doi(a,b);
    writeln(a:6, b:6);
end.
```

Khi chương trình thực hiện, sẽ nhận được kết quả:

```
5 10
5 5
```

2. Cách viết và sử dụng hàm

Điểm khác nhau cơ bản giữa thủ tục và hàm là việc thực hiện hàm luôn trả về giá trị kết quả thuộc kiểu xác định và giá trị đó được gán cho tên hàm.

Hàm có cấu trúc tương tự như thủ tục, tuy nhiên có khác nhau phần đầu. Khai báo phần đầu một hàm như sau:

function *<tên hàm>* [(*<danh sách tham số>*)]: *<kiểu dữ liệu>*;

trong đó, *kiểu dữ liệu* là kiểu dữ liệu của giá trị mà hàm trả về và chỉ có thể là các kiểu *integer*, *real*, *char*, *boolean*, *string*.

Cũng giống như thủ tục, nếu hàm không có tham số hình thức thì không cần *danh sách tham số*.

Khác với thủ tục, trong thân hàm cần có lệnh gán giá trị cho tên hàm:

<tên hàm> := *<biểu thức>*;

Ví dụ 1

Xét chương trình thực hiện việc rút gọn một phân số, trong đó có sử dụng hàm tính ước chung lớn nhất (ƯCLN) của hai số nguyên.

```
program Rutgon_Phanso;
uses crt;
var TuSo, MauSo, a: integer;
function UCLN(x, y: integer): integer; {bat dau ham UCLN}
   var sodu: integer;
   begin
        while y<>0 do
           begin
              sodu:= x mod y;
              x:= y;
              y:= sodu;
           end;
        UCLN:=x;
    end; {het ham UCLN}
begin
   clrscr;
   write('Nhap tu so, mau so vao! '); readln(TuSo, MauSo);
   a:=UCLN(TuSo, MauSo);
```

```
    if a>1 then
        begin
            TuSo:= TuSo div a;
            MauSo:= MauSo div a;
        end;
    writeln(TuSo:5, MauSo:5);
end.
```

Ghi chú: Trong chương trình này, các biến *TuSo*, *MauSo* và *a* là các biến toàn cục, còn biến *sodu* là biến cục bộ.

Sử dụng hàm

Việc sử dụng hàm *hoàn toàn tương tự* với việc sử dụng các hàm chuẩn, khi viết lệnh gọi gồm tên hàm và tham số thực sự tương ứng với các tham số hình thức.

Lệnh gọi hàm có thể tham gia vào biểu thức như một toán hạng và thậm chí là tham số của lời gọi hàm, thủ tục khác, ví dụ:

```
A:= 6*Ucln(TuSo,MauSo)+1;
```

Ví dụ 2

Chương trình sau cho biết giá trị nhỏ nhất trong ba số nhập từ bàn phím, trong đó có sử dụng hàm tìm số nhỏ nhất trong hai số.

```
program Minbaso;
var a, b, c: real;
{Ham tim so nho nhat trong hai so a va b}
function Min(a,b:real):real;
    begin
        if a<b then Min:=a
            else Min:=b;
    end;
begin
    write('Nhap vao ba so: ');
    readln(a,b,c);
    writeln('So nho nhat trong ba so la: ', Min(Min(a,b),c));
    readln
end.
```

Bài tập và thực hành 6

1. Mục đích, yêu cầu

- *Rèn luyện các thao tác xử lí xâu, kĩ năng tạo hiệu ứng chữ chạy trên màn hình;*
- *Nâng cao kĩ năng viết, sử dụng chương trình con.*

2. Nội dung

a) Trước hết, hãy tìm hiểu việc xây dựng hai thủ tục sau đây:

- Thủ tục *CatDan*(*s*1, *s*2) nhận đầu vào là xâu *s*1 gồm không quá 79 kí tự, tạo xâu *s*2 thu được từ xâu *s*1 bằng việc chuyển kí tự đầu tiên của nó xuống vị trí cuối cùng. Ví dụ nếu *s*1 = '*abcd*' thì *s*2 = '*bcda*'.

```
type str79 = string[79];
procedure CatDan(s1: str79; var s2: str79);
     begin
       s2:= copy(s1,2,length(s1)-1)+s1[1];
     end;
```

- Thủ tục *CanGiua*(*s*) nhận đầu vào là xâu *s* gồm không quá 79 kí tự, bổ sung vào đầu *s* một số dấu cách để khi đưa ra màn hình xâu kí tự *s* ban đầu được căn giữa dòng (80 kí tự).

```
procedure CanGiua(var s: str79);
    var i, n: integer;
    begin
     n:= length(s);
     n:= (80-n) div 2;
     for i:= 1 to n do s:= ' '+s;
    end;
```

b) Theo dõi cách sử dụng hai thủ tục trên, ta có thể viết chương trình sau đây để nhập một xâu kí tự từ bàn phím và đưa xâu đó ra màn hình có dạng dòng chữ chạy giữa màn hình văn bản 25×80.

```
uses crt;
type str79 = string[79];
```

```
var s1, s2: str79;
   stop: boolean;
procedure CatDan(s1: str79; var s2: str79);
    begin
      s2:= copy(s1,2,length(s1)-1)+s1[1];
    end;
procedure CanGiua(var s: str79);
    var i, n: integer;
    begin
      n:= length(s);
      n:= (80-n) div 2;
      for i:= 1 to n do s:= ' '+s;
    end;
begin
 clrscr;
 write('Nhap xau s1: '); readln(s1);
 CanGiua(s1);
 clrscr;
 stop:= false;
 while not(stop) do
   begin
      gotoxy(1,12); (* Chuyen con tro den dau dong 12*)
      write(s1);
      delay(500); (* Dung 500 miligiay *)
      CatDan(s1, s2);
      s1:=s2;
      stop:=keypressed; (* Nhan mot phim bat ki de ket thuc*)
   end;
 readln
end.
```

Hãy chạy thử chương trình trên với dòng chữ

'... Mung nghin nam Thang Long - Ha Noi!... '

c) Hãy viết thủ tục *ChuChay*(*s*, *dong*) nhận đầu vào là xâu *s* gồm không quá 79 kí tự và biến nguyên *dong*, đưa ra xâu *s* có dạng chữ chạy ở dòng *dong*. Viết và chạy chương trình có sử dụng thủ tục này.

Bài tập và thực hành 7

1. Mục đích, yêu cầu

- *Nâng cao kĩ năng viết, sử dụng chương trình con;*
- *Biết cách viết một chương trình có cấu trúc để giải một bài toán trên máy tính.*

2. Nội dung

a) Tìm hiểu việc xây dựng các hàm và thủ tục thực hiện tính độ dài các cạnh, chu vi, diện tích, kiểm tra các tính chất đều, cân, vuông của tam giác được trình bày dưới đây.

Giả thiết tam giác được xác định bởi toạ độ của ba đỉnh. Ta sử dụng kiểu bản ghi để mô tả một tam giác:

```
type   Diem = record
                 x, y: real;
              end;
       Tamgiac = record
                    A, B, C: Diem;
                 end;
```

Ta xây dựng các thủ tục và hàm:

- Thủ tục nhận dữ liệu vào là biến mô tả tam giác R và dữ liệu ra là độ dài của ba cạnh a, b, c:

  ```
  procedure Daicanh(var R: Tamgiac; var a, b, c: real);
  ```

- Hàm tính chu vi của tam giác R:

  ```
  function Chuvi(var R: Tamgiac): real;
  ```

- Hàm tính diện tích của tam giác R:

  ```
  function Dientich(var R: Tamgiac): real;
  ```

- Thủ tục nhận đầu vào là biến mô tả tam giác R và đầu ra là tính chất của tam giác (*Deu* hay *Can* hay *Vuong*):

  ```
  procedure Tinhchat(var R:Tamgiac;var Deu,Can,Vuong:boolean);
  ```

- Thủ tục hiển thị toạ độ ba đỉnh tam giác lên màn hình:
  `procedure Hienthi(var R: Tamgiac);`
- Hàm tính khoảng cách giữa hai điểm P, Q:
  `function Kh_cach(P,Q: Diem): real;`

b) Tìm hiểu chương trình nhập vào toạ độ ba đỉnh một tam giác và sử dụng các hàm, thủ tục được xây dựng dưới đây để khảo sát các tính chất của tam giác.

```
uses crt;
const eps = 1.0E-6;
type
   Diem = record
             x, y: real;
          end;
   Tamgiac = record
                A, B, C: Diem;
             end;
var T: Tamgiac;
   Deu, Can, Vuong: boolean;
function Kh_cach(P, Q: Diem): real;
   begin
      Kh_cach:=sqrt((P.x-Q.x)*(P.x-Q.x)+(P.y-Q.y)*(P.y-Q.y));
   end;
procedure Daicanh(var R: Tamgiac; var a, b, c: real);
   begin
      a:= Kh_cach(R.B, R.C);
      b:= Kh_cach(R.A, R.C);
      c:= Kh_cach(R.A, R.B);
   end;
function ChuVi(var R: Tamgiac): real;
   var a, b, c: real;
   begin
      Daicanh(R, a, b, c);
      ChuVi:= a + b + c;
   end;
```

```
function Dientich(var R: Tamgiac): real;
   var a, b, c, p: real;
   begin
      Daicanh(R, a, b, c);
      p:= (a+b+c)/2;
      Dientich:= sqrt(p*(p-a)*(p-b)*(p-c))
   end;
procedure Hienthi(var R: tamgiac);
   begin
      writeln('Toa do 3 dinh cua tam giac la: ');
      writeln(' - Dinh A(',R.A.x:0:3,', ', R.A.y:0:3,')');
      writeln(' - Dinh B(',R.B.x:0:3,', ', R.B.y:0:3,')');
      writeln(' - Dinh C(',R.C.x:0:3,', ', R.C.y:0:3,')');
   end;
 procedure Tinhchat(var R:Tamgiac;var Deu,Can,Vuong:boolean);
    var a, b, c: real;
    begin
      Deu:= false; Can:= false; Vuong:= false;
      Daicanh(R, a, b, c);
      if (abs(a-b)<eps) and (abs(a-c)<eps) then
         Deu:= true
      else
        if (abs(a-b)<eps) or (abs(a-c)<eps) or (abs(b-c)<eps)
        then Can:= true;
      if (abs(a*a+b*b-c*c)<eps) or (abs(a*a+c*c-b*b)<eps)
        or (abs(b*b+c*c-a*a)<eps) then Vuong:= true;
    end;
begin
   writeln('Nhap tam giac: ');
   write('Toa do dinh A: '); readln(T.A.x, T.A.y);
   write('Toa do dinh B: '); readln(T.B.x, T.B.y);
   write('Toa do dinh C: '); readln(T.C.x, T.C.y);
```

```
    writeln('====================================');
    Hienthi(T);
    writeln('Dien tich: ',Dientich(T):9:3);
    writeln('Chu vi: ',Chuvi(T):9:3);
    Tinhchat(T, Deu, Can, Vuong);
    writeln('Tam giac co tinh chat: ');
    if Deu then writeln(' la tam giac deu')
       else if Can then writeln(' la tam giac can');
    if Vuong then writeln(' la tam giac vuong');
    readln;
end.
```

c) Viết chương trình sử dụng các hàm và thủ tục xây dựng ở trên để giải bài toán:

Cho tệp dữ liệu TAMGIAC.DAT có cấu trúc như sau:

- Dòng đầu tiên chứa số N;
- N dòng tiếp theo, mỗi dòng chứa sáu số thực x_A, y_A, x_B, y_B, x_C, y_C là toạ độ ba đỉnh $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$ của tam giác ABC.

Hãy nhập dữ liệu từ tệp đã cho và trong số N tam giác đó, đưa ra tệp TAMGIAC.OUT gồm ba dòng:

- Dòng đầu tiên là số lượng tam giác đều;
- Dòng thứ hai là số lượng tam giác cân (nhưng không là đều);
- Dòng thứ ba là số lượng tam giác vuông.

Bài đọc thêm 3

AI LÀ LẬP TRÌNH VIÊN ĐẦU TIÊN?

Đó là một phụ nữ, bà Ada Augusta Byron Lovelace, con gái của nhà thơ nổi tiếng thời đó Lord Byron. Ada là một trong những nhân vật ấn tượng nhất trong lịch sử Tin học. Bà sinh ngày 10/12/1815 và là người cùng thời với Charles Babbage, người đầu tiên đưa ra đề án thiết kế chiếc máy tính điều khiển theo chương trình có tên là Analytical Engine (máy phân tích).

Ada Augusta Byron Lovelace (1815 – 1852)

Từ nhỏ, bà đã nổi tiếng là một người thông minh, có khả năng đặc biệt về toán học.

Ngay từ khi bản thiết kế máy phân tích còn ở trên giấy, Ada đã đề xuất với Babbage một kế hoạch chi tiết để máy phân tích tính các số Bernoulli. Ngày nay người ta coi kế hoạch này là *chương trình máy tính đầu tiên* và bà được gọi là *lập trình viên đầu tiên*.

Các ghi chép được công bố của Ada cho tới nay vẫn đặc biệt có ý nghĩa đối với các lập trình viên. Giáo sư J. Von Neumann đã viết rằng các quan sát của Ada "chứng tỏ bà đã hiểu được các nguyên tắc lập trình máy tính trước thời đại của mình hàng thế kỉ".

Như một nhà toán học, Ada đánh giá cao khả năng tự động hoá các công việc tính toán nặng nhọc. Nhưng bà quan tâm hơn đến ***các nguyên tắc của việc lập trình*** các thiết bị đó. Ngay khi máy phân tích còn chưa được xây dựng, Ada đã thí nghiệm viết những dãy lệnh. Bà nhận ra giá trị của một vài thủ thuật đặc biệt trong nghệ thuật mới này và điều thú vị là những thủ thuật này hiện giờ vẫn còn là cơ bản đối với các ngôn ngữ lập trình hiện đại, đó chính là ***chương trình con, vòng lặp và các phép chuyển điều khiển***.

Thay cho việc viết các dãy lệnh lặp đi lặp lại nhiều lần, ta có thể viết chúng dưới dạng các ***chương trình con*** để dùng nhiều lần. Các chương trình con ngày nay là thành phần không thể thiếu được của mọi ngôn ngữ lập trình.

Máy phân tích và các máy tính số thực hiện rất tốt các tính toán nhiều lần một cách nhanh chóng. Thời kì đó, các bìa đục lỗ được sử dụng để đưa dữ liệu và các lệnh vào máy. Bằng việc phát minh ra các lệnh thực hiện việc chuyển thiết bị đọc bìa về một bìa xác định trước nó, sao cho dãy các lệnh có thể được thực hiện một số lần nhất định,

Ada đã phát minh ra ***vòng lặp*** – một trong những cấu trúc điều khiển quan trọng trong các ngôn ngữ lập trình.

Khả năng lôgic của Ada đã phát huy với ***phép chuyển điều khiển có điều kiện***. Bà nghĩ ra một loại lệnh để thao tác với thiết bị đọc bìa, nhưng thay cho việc quay lại và lặp lại dãy bìa, lệnh này cho phép thiết bị đọc bìa chuyển tới một bìa khác tại bất kì vị trí nào trong dãy, NẾU một điều kiện nào đó được thoả mãn. Việc thêm chữ NẾU đó vào danh sách các lệnh số học thuần tuý trước đây có nghĩa là chương trình có thể làm nhiều hơn là tính toán đơn thuần. Ở dạng thô sơ nhưng về tiềm năng là rất có ý nghĩa, máy phân tích có thể thực hiện các *quyết định*.

Ada mất năm 1852, khi mới qua tuổi 36. Nếu như bà không qua đời sớm như vậy, chắc chắn khoa học lập trình của thế kỉ XIX đã có thể tiến nhanh hơn nhiều.

Để tưởng nhớ công lao của Ada, một ngôn ngữ lập trình do Bộ Quốc phòng Mĩ tạo ra năm 1979 đã được mang tên bà.

§19. THƯ VIỆN CHƯƠNG TRÌNH CON CHUẨN

Mỗi ngôn ngữ lập trình đều có một số lượng phong phú các chương trình con chuẩn trong các thư viện. Dưới đây giới thiệu sơ lược nội dung của một số thư viện chương trình con chuẩn của Pascal.

1. CRT

Thư viện ***crt*** chứa các thủ tục liên quan đến việc quản lí và khai thác màn hình, bàn phím của máy tính. Dùng các thủ tục của thư viện này, người lập trình có thể điều khiển việc đưa dữ liệu ra màn hình, xây dựng các giao diện màn hình-bàn phím, dùng bàn phím điều khiển chương trình hoặc sử dụng âm thanh để xây dựng các chương trình mô phỏng.

Ngoài thủ tục *clrscr* đã giới thiệu, dưới đây là một số thủ tục tiện ích khác:

Thủ tục *TextColor*(*color*) đặt màu cho chữ trên màn hình, trong đó *color* là hằng hoặc biến xác định màu và có thể nhận giá trị trong bảng sau:

Màu	Tên hằng	Giá trị
Đen	black	0
Xanh trời	blue	1
Xanh lá	green	2
Xanh lơ	cyan	3
Đỏ	red	4
Tím	magenta	5
Vàng	yellow	14
Trắng	white	15

Thủ tục *TextBackground(color)* đặt màu cho nền của màn hình, trong đó *color* là hằng hoặc biến xác định màu và có thể nhận giá trị trong bảng trên.

Thủ tục *GotoXY(x, y)* đưa con trỏ tới vị trí cột x dòng y của màn hình văn bản. Do màn hình văn bản gồm 25 dòng và 80 cột nên phạm vi giá trị của các tham số là $1 \le x \le 80$, $1 \le y \le 25$.

2. GRAPH

Mặc dù Pascal không phải là ngôn ngữ chuyên về đồ hoạ, nhưng với thư viện đồ hoạ người lập trình có thể khai thác khả năng đồ hoạ của máy tính ở những mức độ thông dụng.

Thư viện này chứa các hàm, thủ tục liên quan đến chế độ đồ hoạ của các loại màn hình khác nhau và cho phép thực hiện các thao tác đồ hoạ cơ bản như vẽ điểm, đường, tô màu,...

a) Các thiết bị và chương trình hỗ trợ đồ hoạ

Màn hình có thể làm việc trong hai chế độ: *chế độ văn bản* và *chế độ đồ hoạ*. Có thể hình dung màn hình như một bảng các điểm sáng. Hình ảnh đồ hoạ được xây dựng từ các điểm sáng. Mỗi điểm sáng là một điểm ảnh (pixel) và điểm ảnh là đơn vị cơ sở của màn hình đồ hoạ.

Bảng mạch điều khiển màn hình là thiết bị đảm bảo tương tác giữa bộ xử lí và màn hình để thực hiện các chế độ phân giải và màu sắc. Tên gọi của bảng mạch điều khiển màn hình thường trùng với loại màn hình, ví dụ VGA, SVGA,...

Trong Turbo Pascal, thư viện ***graph*** cung cấp các chương trình điều khiển tương ứng với các loại bảng mạch đồ hoạ. Các chương trình điều khiển này nằm trong các tệp có phần mở rộng là BGI (Borland Graphics Interface). Muốn hoạt động trong chế độ đồ hoạ, cần phải có tệp BGI thích hợp với màn hình đang dùng. Các tệp BGI được Pascal ngầm định để trong thư mục con BGI. Khi khởi động đồ hoạ, cần chỉ rõ đường dẫn đến các tệp này.

Toạ độ trên màn hình đồ hoạ được đánh số từ 0, cột được tính từ trái sang phải và dòng được tính từ trên xuống dưới. Độ phân giải màn hình VGA thường được đặt là 640×480.

b) Khởi tạo chế độ đồ hoạ

Một chương trình đồ hoạ bao giờ cũng mở đầu bằng việc khởi tạo chế độ đồ hoạ. Các thủ tục, hàm của thư viện ***graph*** chỉ hoạt động khi chế độ đồ hoạ đã được thiết lập. Khi kết thúc làm việc với chế độ đồ hoạ thì cần quay về chế độ văn bản.

Thủ tục sau dùng để thiết lập chế độ đồ hoạ:

```
procedure InitGraph(var driver,mode:integer; path:string);
```

trong đó:

– *driver* là số hiệu của trình điều khiển BGI;

– *mode* là số hiệu của độ phân giải;

– *path* là đường dẫn đến các tệp BGI.

Thông thường, ta nên sử dụng cách thiết lập chế độ đồ hoạ tự động với biến *driver* được gán giá trị 0 (được định sẵn bằng hằng *detect*).

Ví dụ

Giả sử màn hình làm việc là VGA và các tệp BGI đang để ở thư mục C:\TP\BGI, khi đó các lệnh sau sẽ thiết lập đồ hoạ với chế độ VGAHi:

```
driver:= 0;
InitGraph(driver, mode, 'C:\TP\BGI');
```

Sau khi kết thúc làm việc với chế độ đồ hoạ, để trở về chế độ văn bản ta gọi thực hiện thủ tục:

```
CloseGraph;
```

c) *Các thủ tục vẽ điểm, đoạn thẳng*

Vẽ điểm và đoạn thẳng là hai thao tác cơ bản của đồ hoạ.

- Trước khi vẽ, ta có thể đặt màu cho nét vẽ bằng thủ tục:

```
procedure SetColor(color: word);
```

- Vẽ điểm được thực hiện bằng thủ tục:

```
procedure PutPixel(x,y: integer; color: word);
```

trong đó:

– *x* và *y* là toạ độ của điểm;

– *color* là màu của điểm.

- Đoạn thẳng được xác định bởi toạ độ của hai điểm đầu, cuối. Để vẽ đoạn thẳng nối hai điểm ta sử dụng thủ tục:

```
procedure Line(x1, y1, x2, y2: integer);
```

trong đó (*x*1, *y*1) và (*x*2, *y*2) là các toạ độ của hai điểm đầu, cuối của đoạn thẳng.

- Vẽ đoạn thẳng nối điểm hiện tại (vị trí con trỏ) với điểm có toạ độ (*x*, *y*):

```
procedure LineTo(x, y: integer);
```

- Vẽ đoạn thẳng nối điểm hiện tại với điểm có toạ độ bằng toạ độ hiện tại cộng với gia số (*dx*, *dy*):

```
procedure LineRel(dx, dy: integer);
```

d) *Các thủ tục và hàm liên quan đến vị trí con trỏ*

- Các hàm xác định giá trị lớn nhất có thể của toạ độ màn hình *X* và *Y* (để biết độ phân giải màn hình trong chế độ đồ hoạ đang sử dụng):

```
function GetMaxX: integer;
function GetMaxY: integer;
```

- Thủ tục chuyển con trỏ tới toạ độ (*x*, *y*):

```
procedure MoveTo(x, y: integer);
```

e) *Một số thủ tục vẽ hình đơn giản*

- Vẽ đường tròn có tâm tại (*x*, *y*), bán kính *r*:

```
procedure Circle(x,y: integer; r: word);
```

- Vẽ cung của elip có tâm tại điểm (*x, y*) với các bán kính trục *Xr*, *Yr* từ góc khởi đầu *StAngle* đến góc kết thúc *EndAngle*:

```
procedure Ellipse(x,y:integer;StAngle,EndAngle,Xr,Yr: word);
```

- Vẽ hình chữ nhật có các cạnh song song với các trục toạ độ, (*x*1, *y*1) là toạ độ của đỉnh trái trên còn (*x*2, *y*2) là toạ độ của đỉnh phải dưới:

```
procedure Rectangle(x1,y1,x2,y2: integer);
```

3. Một số thư viện khác

System: Trong thư viện chuẩn này chứa các hàm sơ cấp và các thủ tục vào/ra mà các chương trình đều dùng tới.

Dos: Thư viện này chứa các thủ tục cho phép thực hiện trực tiếp các lệnh như tạo thư mục, thiết lập giờ hệ thống,...

Printer: Thư viện này cung cấp các thủ tục làm việc với máy in.

4. Sử dụng thư viện

Muốn sử dụng các thủ tục và hàm chuẩn của một (một số) thư viện nào đó (trừ *system*) phải dùng lệnh khai báo:

```
uses unit1,unit2,..., unitN;
```

trong đó, ***uses*** là từ khoá, *unit*1, *unit*2, ..., *unitN* là tên các thư viện (được viết cách nhau bởi dấu phẩy).

Khai báo này phải là lệnh đầu tiên trong phần khai báo (nghĩa là nó chỉ viết sau khai báo tên chương trình).

Ví dụ

Để sử dụng các hàm và thủ tục chuẩn trong các thư viện *crt*, *dos*, *graph*, ta cần khai báo:

```
uses crt, dos, graph;
```

Bài tập và thực hành 8

1. Mục đích, yêu cầu

Giới thiệu một số chương trình để học sinh thấy được khả năng đồ hoạ của Pascal.

2. Nội dung

a) Chương trình sau đây vẽ các đường gấp khúc "ngẫu nhiên" nhờ thủ tục *LineTo*, mỗi đoạn có một màu ngẫu nhiên. Vị trí bắt đầu vẽ là tâm của màn hình. Kết thúc việc vẽ bằng cách nhấn một phím bất kì. Chạy thử chương trình và quan sát kết quả trên màn hình.

```
uses crt,graph;
var stop: boolean;
function DetectInit(path: string): integer;
   var drive, mode: integer;
   begin
      drive:= 0;
      InitGraph(drive, mode, path);
      DetectInit:= GraphResult;
   end;
begin
   if DetectInit('C:\TP\BGI') <> 0 then
      begin
         write('Loi do hoa! Nhan phim Enter de ket thuc...');
         readln;
      end
   else
      begin
         Randomize;
         MoveTo(Getmaxx div 2,Getmaxy div 2);
         stop:=false;
         while not(stop) do
```

```
            begin
                SetColor(Random(GetMaxColor));
                {Thiet lap mau mot cach ngau nhien}
                LineTo(Random(Getmaxx),Random(Getmaxy));
                Delay(200);  {Tam dung}
                stop:= Keypressed;
            end;
        end;
    CloseGraph
end.
```

b) Chương trình dưới đây minh hoạ việc sử dụng các thủ tục vẽ hình đơn giản. Hãy chạy chương trình rồi thay đổi một số tham số như màu vẽ, toạ độ và quan sát kết quả trên màn hình.

```
program GraphDemo;
uses graph;
var
    gd, gm: integer;
    xm, ym, xmaxD4, ymaxD4: word;
begin
    gd:= detect;
    Initgraph(gd, gm, 'C:\TP\BGI');
    xm:=GetmaxX div 2; ym:= GetmaxY div 2;
    {Ve hinh chu nhat voi net ve mau vang}
    SetColor(Yellow);
    Rectangle(10,10,xm,ym);
    readln;
    {Ve duong tron mau xanh la cay, tam(450;100) ban kinh 50}
    SetColor(LightGreen);
    Circle(450, 100, 50);
    readln;
    {Ve ellip mau do }
    SetColor(Red);
    Ellipse(100, 200, 0, 360, 50,120);
    readln;
    CloseGraph
end.
```

TÓM TẮT

- Chương trình con đóng vai trò quan trọng trong lập trình, đặc biệt trong lập trình có cấu trúc.
- Dùng chương trình con sẽ thuận lợi cho việc tổ chức, viết, kiểm tra chương trình và sử dụng lại.
- Chương trình con có phần đầu, phần khai báo và phần thân.
- Chương trình con có thể có *tham số hình thức* khi khai báo và được thay bằng *tham số thực sự* khi gọi. Các tham số hình thức và thực sự phải tương ứng về thứ tự và kiểu dữ liệu.
- Chương trình con được gọi bằng tên của nó.
- Biến được khai báo trong chương trình con là *biến cục bộ*.
- Thư viện cung cấp những chương trình con chuẩn mở rộng khả năng ứng dụng.

CÂU HỎI VÀ BÀI TẬP

1. Hãy nêu sự giống nhau và khác nhau giữa thủ tục và hàm.
2. Chương trình con có thể không có tham số được không? Cho ví dụ.
3. Hãy cho ví dụ chương trình con có nhiều hơn một kết quả ra.
4. Viết chương trình con (hàm, thủ tục) tính bội số chung nhỏ nhất của hai số nguyên dương *a*, *b*. Hãy cho biết trong trường hợp này viết chương trình con dưới dạng hàm hay thủ tục là thuận tiện hơn ? Vì sao?

Bài đọc thêm 4

ÂM THANH

Ta sẽ tìm hiểu thêm một số yếu tố về âm thanh như mô phỏng âm thanh và mô phỏng nốt nhạc trong Pascal.

Khi được bật, loa máy tính sẽ phát ra âm thanh ở một tần số nào đó cho đến khi được tắt. Để khai thác khả năng này, thư viện **crt** có:

– Thủ tục phát ra một âm thanh có tần số (cao độ) là *h* (tính theo đơn vị héc - Hz):

```
procedure Sound(h: word);
```

– Thủ tục tắt âm thanh đang được phát ra loa:

```
procedure NoSound;
```

Mỗi khi gọi thủ tục *Sound* để phát một âm thanh nào đó, âm thanh này sẽ được kéo dài cho đến khi gặp lời gọi *NoSound*. Để xác định khoảng thời gian kéo dài (trường độ), ta dùng thủ tục *Delay* với tham trị *t*. Thủ tục sau phát một âm thanh với cao độ *h* và trường độ *t*:

```
procedure CreatSound(h,t: word);
    begin
       Sound(h); Delay(t); NoSound;
    end;
```

a) *Mô phỏng âm thanh*

Có thể mô phỏng một âm thanh nhờ thủ tục *CreatSound(h,t)*. Vấn đề còn lại là tìm các giá trị thích hợp của *h* và *t*.

Ví dụ

Chương trình dưới đây tạo tiếng lộc cộc của vó ngựa phi:

```
program DemoSound;
uses crt;
procedure CreatSound(h, t: word);
    begin
      Sound(h); Delay(t); NoSound;
    end;
```

```
procedure Clockoc(h: word);
    begin
      CreatSound(h, 20);
      Delay(50);
      CreatSound(h, 20);
      Delay(100);
      CreatSound(h, 40);
    end;
begin
   repeat
      Clockoc(500);
      Delay(200);
   until KeyPressed;
end.
```

Việc khai thác âm thanh thường được sử dụng để xây dựng các chương trình trò chơi hoặc mô phỏng. Vì thế, nếu thường xuyên phải phát triển những ứng dụng loại này thì nên tổ chức một thư viện chứa một số âm thanh mẫu như tiếng các loại động cơ, tiếng còi, tiếng nổ, tiếng va chạm,... để tiện dùng.

b) Mô phỏng nốt nhạc

Một trường hợp riêng của việc khai thác âm thanh là mô phỏng các nốt nhạc. Một nốt nhạc cao độ *h*, trường độ *t* sẽ được thực hiện bởi lệnh gọi:

```
CreatSound(h,t);
```

Theo âm luật, từ một quãng 8 đến quãng 8 kế tiếp, thừa số nhân của cao độ là 2. Trong một quãng 8, có 12 khoảng nửa cung, vì thế thừa số nhân giữa hai khoảng này là căn bậc 12 của 2.

Vì vậy chỉ cần biết cao độ của một nốt làm gốc, ta có thể tính cao độ của các nốt khác. Chẳng hạn, xuất phát từ cao độ của nốt đô trung là 512, cao độ của các nốt trong quãng 8 trung lần lượt là (sau khi đã làm tròn):

Nốt nhạc	Cao độ
đô trung	512
đô thăng	542
rê	575
rê thăng	609
mi	645
fa	683

Nốt nhạc	Cao độ
fa thăng	724
son	767
son thăng	813
la	861
la thăng	912
si	967

Về trường độ, lấy nốt móc đơn làm đơn vị là 150, ta tính được các trường độ khác:

Nốt nhạc	Trường độ
nốt đen	300
nốt trắng	600
nốt trắng chấm	900
nốt tròn	1200
...	

Ví dụ

Chương trình dưới đây thể hiện một bản nhạc ngắn của đồng hồ, tất cả các nốt đều ở quãng 8 trung:

```
program DemoMusic;
uses crt;
procedure CreatSound(h, t: word);
   begin
      Sound(h); Delay(t); NoSound;
   end;
begin
   CreatSound(967, 300); {si den}
   CreatSound(767, 300); {son den}
   CreatSound(861, 300); {la den}
   CreatSound(575, 900); {re trang cham}
   Delay(60);
   CreatSound(575, 300); {re den}
   CreatSound(861, 300); {la den}
   CreatSound(967, 300); {si den}
   CreatSound(767, 900); {son trang cham}
   Delay(60);
   CreatSound(967, 300); {si den}
   CreatSound(767, 300); {son den}
   CreatSound(861, 300); {la den}
   CreatSound(575, 900); {re trang cham}
   Delay(60);
   CreatSound(575, 300); {re den}
   CreatSound(861, 300); {la den}
   CreatSound(967, 300); {si den}
   CreatSound(767, 900); {son trang cham}
end.
```

PHỤ LỤC A

1. Một số phép toán thường dùng

Tên phép toán	Kí hiệu toán học	Kí hiệu trong Pascal	Kí hiệu trong C++	Loại phép toán
Cộng	+	+	+	Số học
Trừ	-	-	-	
Nhân	×	*	*	
Chia	/	/	/	
Chia nguyên		div	/	
Lấy phần dư	mod	mod	%	
Nhỏ hơn	<	<	<	Quan hệ
Nhỏ hơn hoặc bằng	≤	<=	<=	
Lớn hơn	>	>	>	
Lớn hơn hoặc bằng	≥	>=	>=	
Bằng	=	=	==	
Khác	≠	<>	!=	
Phủ định	¬	not	!	Lôgic
Hoặc (tuyển)	∨	or	||	
Và (hội)	∧	and	&&	

2. Giá trị phép toán lôgic

Giả thiết các biến *A* và *B* là những biến lôgic. Để thuận tiện, ta sẽ kí hiệu giá trị lôgic *true* là 1 và *false* là 0. Khi đó, kết quả của các phép toán lôgic ***not*** *A*, *A* ***and*** *B* và *A* ***or*** *B* được xác định trong các bảng sau:

A	0	1
not A	1	0

A	0	0	1	1
B	0	1	0	1
A and B	0	0	0	1
A or B	0	1	1	1

PHỤ LỤC B

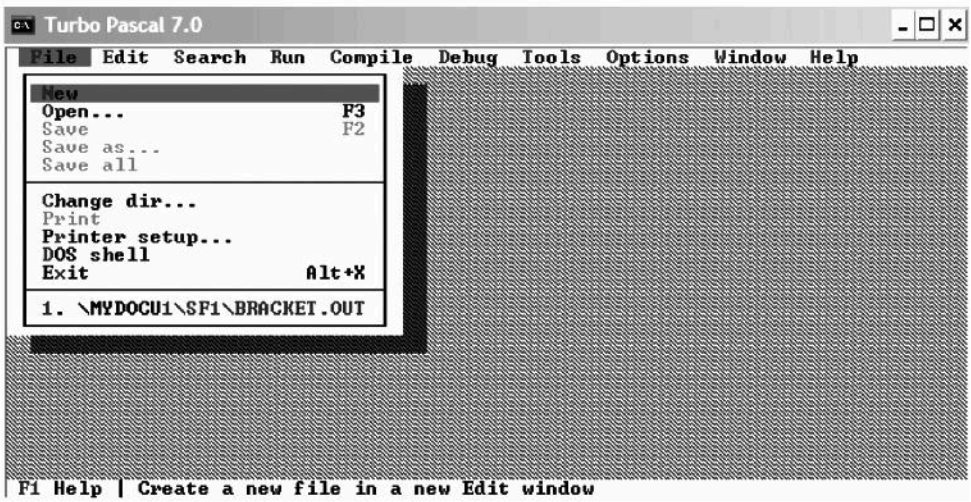
Sau đây là các phụ lục liên quan đến việc sử dụng Pascal.

1. Môi trường Turbo Pascal

Bảng chọn (menu) của Turbo Pascal (TP) được kích hoạt bằng cách nhấn phím **F10**. Khi gõ các phím mũi tên → hay ← ta có thể chuyển tới các mục khác nhau của bảng chọn. Khi nhấn phím **Enter**, bảng chọn sẽ được kích hoạt, giới thiệu các mục công việc. Các bảng chọn thường được dùng là **F**ile (Tệp), **R**un (Thực hiện), **D**ebug (Gỡ rối) và **O**ptions (Tuỳ chọn).

a) Bảng chọn File

Khi kích hoạt bảng chọn này ta có màn hình như hình P.1:



Hình P.1. *Bảng chọn File*

Các lệnh trong bảng chọn này gồm:

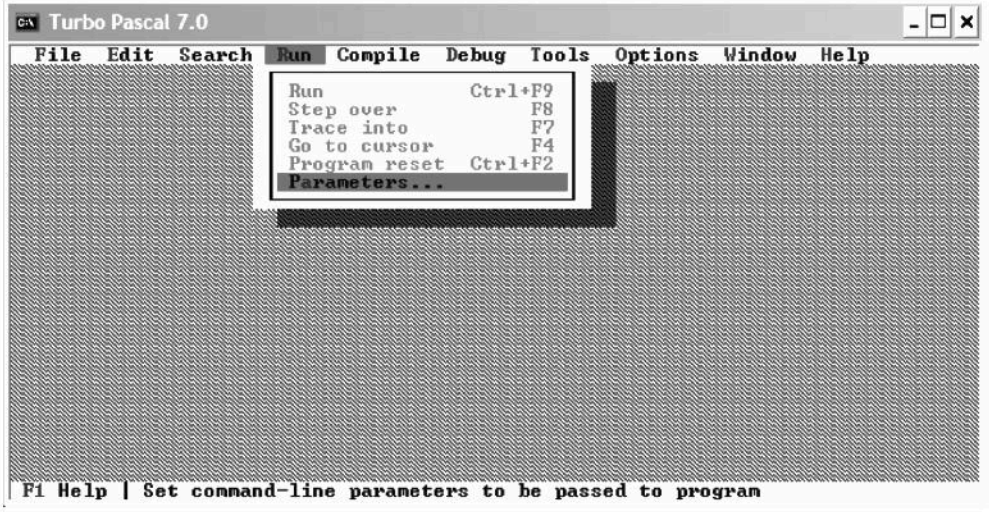
- ***N**ew* - Mở cửa sổ mới để soạn thảo chương trình.
- ***O**pen* - Mở tệp đã có trên đĩa. Có thể thực hiện lệnh này bằng cách nhấn phím **F3**. Trên màn hình sẽ xuất hiện cửa sổ để xác định tên tệp cần mở.
- ***S**ave* - Lưu tệp đang soạn thảo. Nếu đây là tệp chưa đặt tên thì TP sẽ hỏi tên tệp để lưu trữ. Có thể thực hiện lệnh này bằng cách nhấn phím **F2**.
- *Save **a**s* - Lưu tệp đang soạn thảo với tên mới.
- *Save **a**ll* - Lưu tất cả các tệp đang mở.
- ***C**hange dir* - Thay đổi thư mục làm việc.
- ***D**OS shell* - Chuyển sang màn hình DOS, ở đó có thể thực hiện các lệnh của MS-DOS. Để quay trở về màn hình TP cần gõ lệnh EXIT.

- *Exit* - Thoát khỏi TP. Có thể thoát khỏi TP bằng cách nhấn tổ hợp phím **Alt+X**.

Trong một số trường hợp cụ thể, có thể có những mục chưa được phép chọn. Những mục này sẽ hiển thị dưới dạng mờ hơn các mục có thể chọn.

b) Bảng chọn Run

Khi kích hoạt bảng chọn này ta có màn hình tương tự hình P.2. Bảng chọn này dùng để xác định chế độ thực hiện chương trình.



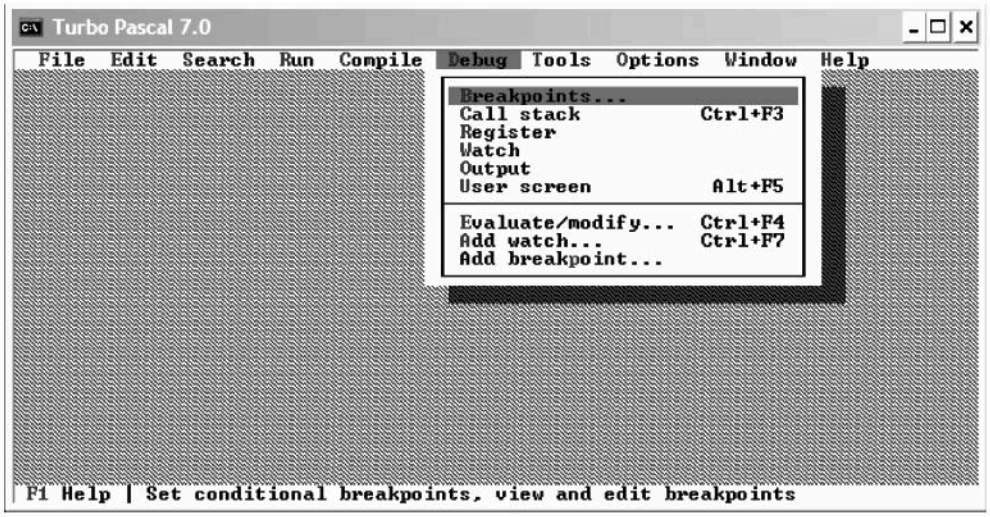
Hình P.2. *Bảng chọn Run*

Các lệnh có thể chọn là:

- ***R**un* - Thực hiện chương trình, nếu trước đó chương trình đã được dịch, bằng cách nhấn tổ hợp phím **Alt+F9** và chương trình không có lỗi. Nếu chương trình chưa được dịch, TP sẽ dịch và thực hiện chương trình khi chương trình nguồn không có lỗi cú pháp. Nếu chương trình nguồn có lỗi cú pháp, TP sẽ thông báo lỗi và con trỏ màn hình sẽ nhấp nháy ở dòng có lỗi. Có thể nhấn tổ hợp phím **Ctrl+F9** để truy cập nhanh lệnh này.
- ***S**tep over* - Thực hiện từng dòng lệnh trên màn hình soạn thảo. Các lời gọi chương trình con được xem như một lệnh. Cần lưu ý là một dòng lệnh trên màn hình soạn thảo có thể chứa *nhiều câu lệnh* của TP. Có thể chọn nhanh lệnh này bằng cách nhấn phím **F8**.
- ***T**race into* - Thực hiện từng dòng lệnh trên màn hình soạn thảo, kể cả các dòng lệnh ở chương trình con. Có thể chọn nhanh lệnh này bằng cách nhấn phím **F7**.
- ***G**o to cursor* - Thực hiện chương trình cho đến dòng có con trỏ màn hình. Có thể chọn nhanh lệnh này bằng cách nhấn phím **F4**.
- ***P**rogram reset* - Xoá các trạng thái của TP đối với chương trình đang thực hiện để chuẩn bị dịch và thực hiện lại từ đầu. Có thể chọn nhanh lệnh này bằng cách nhấn tổ hợp phím **Ctrl+F2**.

c) *Bảng chọn* Debug

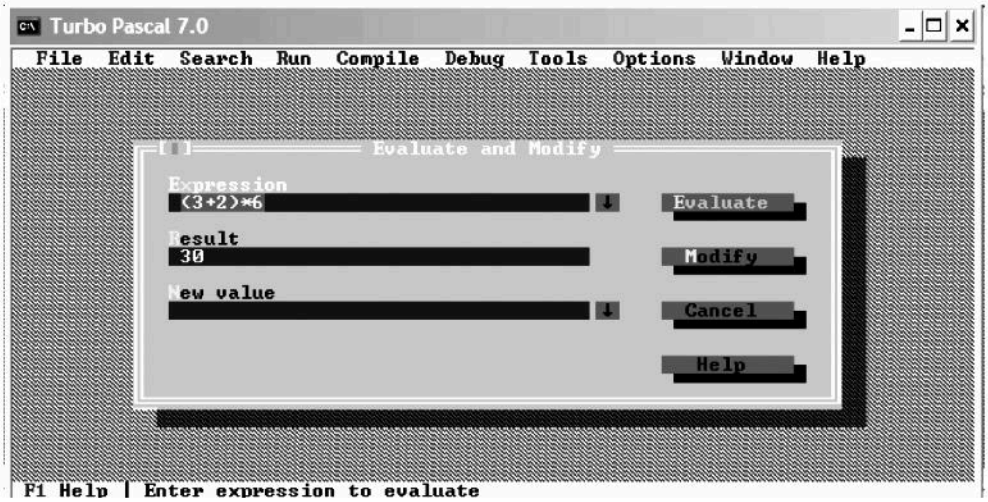
Bảng chọn này dùng để hiệu chỉnh chương trình. Khi kích hoạt bảng chọn này ta có màn hình tương tự hình P.3.



Hình P.3. *Bảng chọn Debug*

Các lệnh thường dùng là:

- ***E**valuate/modify* - Dùng để tính giá trị biểu thức. TP sẽ mở cửa sổ để gõ biểu thức cần tính. Sau khi gõ biểu thức, nhấn phím **Enter**, TP sẽ tính và cho giá trị ở cửa sổ giá trị (h. P.4).

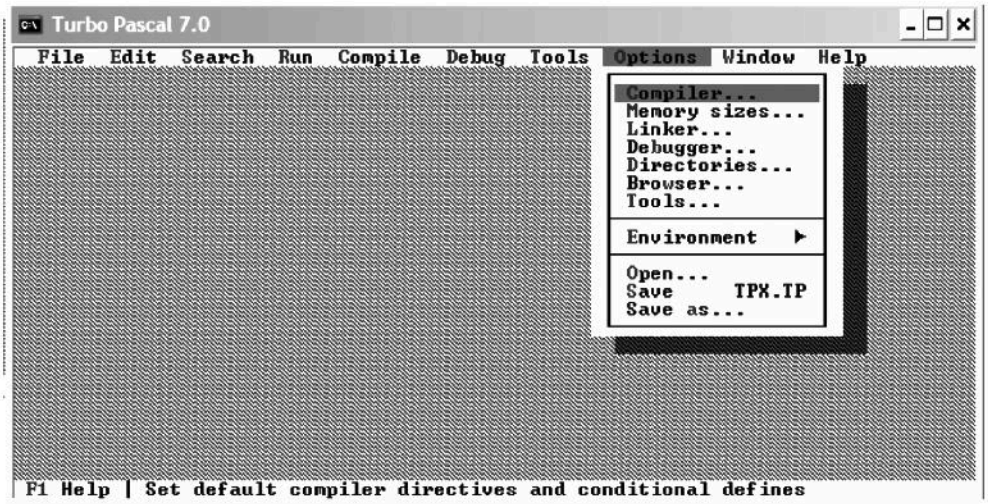


Hình P.4. *Cửa sổ tính giá trị biểu thức*

- ***A**dd watch* - Dùng để mở cửa sổ theo dõi giá trị biến trong quá trình thực hiện chương trình. Có thể chọn nhanh lệnh này bằng cách nhấn tổ hợp phím **Ctrl+F7**.

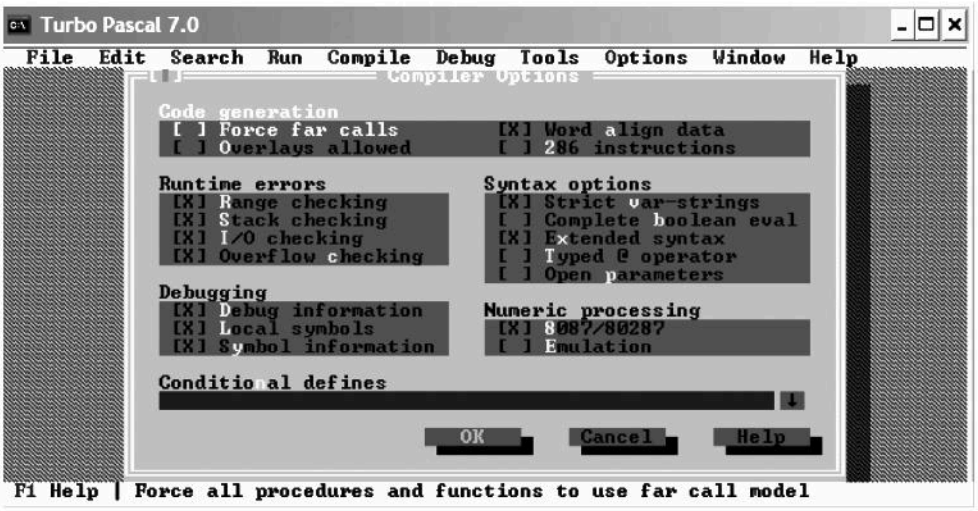
d) ***Bảng chọn* Options**

Bảng chọn này được dùng để đặt các tuỳ chọn cho môi trường lập trình. Mục thường dùng hơn cả là ***C****ompiler* (h. P.5).



Hình P.5. *Bảng chọn Options*

Khi chọn mục ***C****ompiler*, màn hình tương tự như hình P.6 xuất hiện. Ta có thể xác lập các tuỳ chọn cho chương trình dịch. Các tuỳ chọn này cũng có thể xác lập ngay trong chương trình nguồn bằng câu lệnh của TP. Các tuỳ chọn được chia thành nhiều nhóm. Việc chuyển từ nhóm này sang nhóm khác được thực hiện bằng cách nhấn phím **Tab**. Việc chọn hay không chọn một tuỳ chọn được thực hiện bằng cách nhấn *phím cách*. Những tuỳ chọn có hiệu lực được đánh dấu **X** (h. P.6).



Hình P.6. *Các tuỳ chọn cho chương trình dịch*

e) *Chuyển cửa sổ màn hình soạn thảo*

TP cho phép mở nhiều cửa sổ soạn thảo đồng thời. Để chuyển tới cửa sổ tiếp theo, nhấn phím **F6**. Để quay về cửa sổ trước đó, nhấn tổ hợp phím **Shift+F6**.

f) *Trợ giúp*

Để TP có thể đưa ra các trợ giúp, cần có tệp **turbo.tph**. Có thể xem trợ giúp nhờ bảng chọn **Help** hoặc nhấn phím **F1**. TP sẽ hiển thị các mục cho phép ta chọn tiếp vấn đề quan tâm.

Nếu cần hiểu một từ khoá hay tên chuẩn kĩ hơn, ta có thể để con trỏ màn hình tại vị trí trong hoặc cuối tên đó rồi nhấn tổ hợp phím **Ctrl+F1**.

Có thể tra cứu bằng cách nhấn tổ hợp phím **Shift+F1** rồi lần lượt gõ tiếp các kí tự của đối tượng cần tìm hiểu cho đến khi TP tìm được đối tượng đó.

Các phím phục vụ soạn thảo có thể tra cứu bằng cách nhấn phím **F1** vào trợ giúp. Cuối nội dung thông tin trợ giúp này sẽ có danh sách các mục có thể tra cứu về cách soạn thảo (h. P.7).

Hình P.7. *Các mục trợ giúp soạn thảo*

Nhấn phím **Esc** để thoát khỏi trợ giúp.

g) *Các hệ thống phục vụ lập trình trên Pascal*

Trước hết phải kể đến các bộ dịch Pascal của hãng Borland dưới tên gọi Turbo Pascal trong những thập niên cuối thế kỉ XX.

- **Turbo Pascal v1.0** là bộ dịch đầu tiên của hãng Borland. Ngày phát hành: 20/11/1983.

- **Turbo Pascal v5.5** là bước tiến lớn trong phát triển các bộ dịch Pascal của hãng Borland. Bắt đầu từ phiên bản này trong Pascal xuất hiện lập trình hướng đối tượng. Ngày phát hành: 02/05/1989.
- **Borland Pascal with Objects v7.0** (Borland Pascal v7.0) là bộ dịch Pascal nổi tiếng trong những năm cuối của thế kỉ XX và cũng là phiên bản cuối cùng của Turbo Pascal. Ngày phát hành: 09/03/1993.

Các bộ dịch trên có thể tải về từ các trang web:

http://dn.codegear.com/museum hoặc http://pascal.sources.ru/museum/index.htm

Có thể nói các bộ dịch Pascal cho MS-DOS mà đại diện cuối cùng và nổi bật nhất là Borland Pascal v7.0 có thể đáp ứng những yêu cầu cơ bản về giảng dạy lập trình có cấu trúc trên ngôn ngữ lập trình bậc cao. Với sự phát triển mạnh mẽ của hệ điều hành và phần cứng máy tính, công cụ lập trình phải được cải tiến không ngừng để tương thích với hệ điều hành và tận dụng khả năng của phần cứng. Chỉ hỗ trợ hệ điều hành DOS và dành cho Windows một phần khiêm tốn, Turbo Pascal và Borland Pascal đã không đáp ứng được yêu cầu của người lập trình. Một trong những hạn chế cơ bản của các bộ dịch này là chúng là các bộ dịch 16 bit. Hạn chế 64KB cho tất cả các biến và dữ liệu thực sự đã là cản trở lớn cho người lập trình.

Mặc dù Borland không còn quan tâm đến các bộ dịch Turbo Pascal (họ đã xếp chúng vào bảo tàng và chuyển sang phát triển các bộ dịch mới), nhưng những người yêu thích ngôn ngữ này vẫn tiếp tục phát triển Pascal. Dưới đây ta nêu vài bộ dịch 32 bit của Pascal.

- **GPC** (GNU Pascal Compiler) – bộ dịch Pascal 32/64-bit mã nguồn mở, hoàn toàn tương thích với Borland Pascal 7.0, trên nhiều hệ điều hành như Linux (Intel, Alpha), FreeBSD, NetBSD, OpenBSD, DOS 32 bits, MS-Windows 9x/NT, OS/2, MIPS-SGI-IRIX, Alpha-DEC-OSF, Sparc-Sun-Solaris, HP/UX. Thông tin chi tiết về bộ dịch này có thể xem trên trang

  http://www.gnu-pascal.de/gpc/h-about.html
- **TMT** Pascal – bộ dịch 32 bit của Pascal đối với DOS32, Win32 và OS/2. Phiên bản mới nhất là TMT Pascal 5.0 phát hành năm 2003 (xem thông tin trên trang web http://www.tmt.com). Phiên bản 3.90 được phép sử dụng miễn phí, có thể tải về từ địa chỉ http://pascal.sources.ru/tmt/download.htm.
- **FPC** (Free Pascal) có khả năng biên dịch các chương trình 32 bit, tức là chương trình có khả năng quản lí và xử lí dữ liệu được định vị trên 32 bit địa chỉ. Chuẩn lập trình của Free Pascal tương thích hoàn toàn với Turbo Pascal và Borland

Pascal 7.0. Điểm nổi bật nhất của Free Pascal là có khả năng tạo ra các chương trình chạy trên nhiều hệ điều hành khác nhau như DOS, Linux, Win32, OS/2, BeOS, FreeBSD/ELF, AmigaOS, QNX và Solaris. Free Pascal là phần mềm mã nguồn mở. Đây là bộ dịch Pascal được thường xuyên nâng cấp nhất hiện nay. Phiên bản chạy ổn định là 2.0.4 (tháng 8 năm 2006) có thể tải về từ địa chỉ http://www.freepascal.org.

2. Một số tên dành riêng

absolute	external	mod	shr
and	file	nil	string
array	for	not	then
begin	forward	object	to
case	function	of	type
const	goto	or	unit
constructor	if	packed	until
destructor	implementation	procedure	uses
div	in	program	var
do	inline	record	virtual
downto	interface	repeat	while
else	interrupt	set	with
end	label	shl	xor

3. Một số kiểu dữ liệu chuẩn

Kiểu	Loại giá trị	Bộ nhớ (byte)	Phạm vi giá trị
`byte`	Nguyên	1	từ 0 đến 255
`integer`	Nguyên	2	từ –32768 đến 32767
`word`	Nguyên	2	từ 0 đến 65535
`longint`	Nguyên	4	từ –2147483648 đến 2147483647
`real`	Thực	6	0 hoặc có giá trị tuyệt đối trong khoảng $2,9\times10^{-39}$ đến $1,7\times10^{38}$
`char`	Kí tự	1	256 kí tự trong bộ mã ASCII
`boolean`	Lôgic	1	true, false

4. Một số thủ tục và hàm chuẩn

a) Nhóm thủ tục và hàm chuẩn đối với các biến kiểu nguyên

- *Nhóm thủ tục chuẩn*

Thủ tục	Chức năng
`Inc(x)`	Tăng giá trị của biến *x* một đơn vị.
`Dec(x)`	Giảm giá trị của biến *x* một đơn vị.
`Inc(x,y)`	Đặt cho biến *x* giá trị mới bằng giá trị cũ cộng với giá trị của biến *y*.
`Dec(x,y)`	Đặt cho biến *x* giá trị mới bằng giá trị cũ trừ đi giá trị của biến *y*.

- *Nhóm hàm chuẩn*

Hàm	Chức năng
`Sqr(x)`	Cho giá trị bằng bình phương của *x*.
`Pred(x)`	Cho giá trị bằng *x* - 1.
`Succ(x)`	Cho giá trị bằng *x* + 1.
`Random(N)`	Hàm có biểu thức *N* kiểu *word* và cho giá trị là một số nguyên ngẫu nhiên trong phạm vi từ 0 đến *N* - 1. Khi dùng hàm này ta phải gọi thủ tục *randomize*.

b) Nhóm hàm chuẩn đối với các biến kiểu thực

Hàm	Chức năng
`Abs(x)`	Cho giá trị bằng trị tuyệt đối của giá trị biến *x* hoặc số thực *x*.
`ArcTan(x)`	Cho giá trị là số đo của cung thuộc khoảng $(-\frac{\pi}{2}, \frac{\pi}{2})$ có tang bằng giá trị của biến *x* hay số thực *x*.
`Exp(x)`	Cho giá trị bằng luỹ thừa cơ số *e* của giá trị biến *x* hoặc số thực *x*.
`Ln(x)`	Cho giá trị bằng lôgarit cơ số *e* của giá trị biến *x* hoặc số thực *x*.
`Sin(x)`	Cho giá trị bằng sin*x*.
`Cos(x)`	Cho giá trị bằng cosin*x*.
`Pi`	Cho giá trị của số π (3,1415...).
`Int(x)`	Cho giá trị bằng phần nguyên nhưng có kiểu số thực của giá trị biến *x* hoặc số thực *x* (phần nguyên của số thực *x* bằng số nguyên lớn nhất không vượt quá *x*).
`Sqr(x)`	Cho giá trị bằng bình phương của giá trị biến *x* hoặc số thực *x*.
`Sqrt(x)`	Cho giá trị bằng căn bậc hai của giá trị không âm của biến *x* hoặc số thực không âm *x*.
`Randomize`	Thủ tục khởi động sinh số ngẫu nhiên.
`Random`	Cho một số thực ngẫu nhiên trong khoảng (0, 1). Khi dùng hàm này ta phải gọi thủ tục *randomize*.
`Round(x)`	Cho giá trị bằng số nguyên gần số thực *x* nhất *nhưng có kiểu là kiểu số nguyên*. Trong trường hợp phần thập phân của *x* lớn hơn hoặc bằng 0,5 thì hàm cho giá trị làm tròn lên.
`Trunc(x)`	Cho giá trị bằng phần nguyên của *x*.

c) Hàm chuẩn trả về giá trị lôgic

Hàm	Chức năng
`Odd(x)`	Với biểu thức số nguyên *x*, cho giá trị *true* nếu *x* lẻ và cho giá trị *false* nếu *x* chẵn.

d) Nhóm thủ tục và hàm chuẩn đối với biến kiểu kí tự

- *Nhóm thủ tục chuẩn*

Thủ tục	Chức năng
`Inc(x)`	Cho giá trị của biến *x* là kí tự đứng ngay sau kí tự ứng với giá trị hiện thời của *x* trong bộ mã ASCII.
`Dec(x)`	Cho giá trị của biến *x* là kí tự đứng ngay trước kí tự ứng với giá trị hiện thời của *x* trong bộ mã ASCII.

- *Nhóm hàm chuẩn*

Hàm	Chức năng
`Chr(x)`	Cho giá trị là kí tự có mã ASCII thập phân bằng (giá trị của biểu thức) *x* có giá trị nguyên từ 0 đến 255.
`Ord(ch)`	Cho giá trị mã ASCII thập phân của kí tự *ch*.
`Pred(ch)`	Cho kí tự đứng ngay trước kí tự *ch* trong bộ mã ASCII.
`Succ(ch)`	Cho kí tự đứng ngay sau kí tự *ch* trong bộ mã ASCII.
`UpCase(ch)`	Nếu *ch* là chữ cái tiếng Anh, hàm cho giá trị bằng chữ cái hoa tương ứng, ngược lại, hàm cho giá trị bằng giá trị của *ch*.

5. Câu lệnh rẽ nhánh và lặp

a) Câu lệnh case-of

Xét bài toán: Lập chương trình nhập từ bàn phím tháng và năm rồi tính và đưa ra màn hình số ngày của tháng.

Câu lệnh *if-then* giải quyết việc rẽ nhánh tuỳ theo một trong hai khả năng. Bài toán trên liên quan tới việc chọn lựa một trong nhiều khả năng (rẽ nhiều nhánh). Khi đó, nếu dùng câu lệnh *if-then* giải quyết sẽ làm chương trình rườm rà. Câu lệnh chọn *case-of* trong TP cho phép rẽ nhiều nhánh một cách thuận lợi.

Câu lệnh *case-of* có dạng:

`case` <*biểu thức nguyên hoặc kí tự*> `of`

<*danh sách 1*>: <*câu lệnh 1*>;

```
        <danh sách 2>: <câu lệnh 2>;
        . . . . . .
        <danh sách N>: <câu lệnh N>
        [else <câu lệnh N + 1>]
    end;
```

trong đó:

- *Danh sách i* (*i* = 1, 2,..., *N*) là một hoặc nhiều giá trị của biểu thức nêu sau *case*, nếu có nhiều giá trị thì các giá trị viết cách nhau bởi dấu phẩy.
- Khi thực hiện, biểu thức sau từ khoá *case* sẽ được tính, giá trị nhận được sẽ lần lượt được kiểm tra xem nằm trong danh sách nào. Câu lệnh tương ứng với danh sách đầu tiên tìm thấy sẽ được thực hiện. Tiếp theo, thực hiện câu lệnh sau câu lệnh *case-of*.
- Nếu giá trị tính được không xuất hiện ở bất kì danh sách nào thì *câu lệnh N + 1* sau *else* (nếu có) sẽ được thực hiện.

Chương trình để giải bài toán vừa nêu như sau:

```
program Vi_du_case;
uses  crt;
var  T, N, SN: integer;
begin
    clrscr;
    write('Cho biet thang va nam: ');
    readln(T,N);
    case  T  of
        4,6,9,11: SN:= 30;
        2: if (N mod 400=0) or ((N mod 100<>0) and (N mod 4=0))
                then SN:= 29 else SN:= 28
        else SN:= 31
   end;
   writeln('Thang ', T,' nam ', N, ' co ', SN,' ngay.');
   readln
end.
```

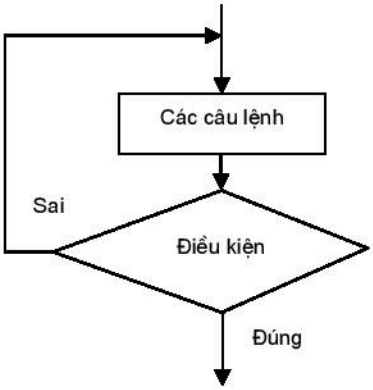
b) *Câu lệnh lặp repeat-until*

Câu lệnh này dùng để tổ chức lặp với số lần lặp không biết trước. Khác với câu lệnh *while-do*, điều kiện kết thúc lặp trong câu lệnh *repeat-until* được kiểm tra sau và có dạng:

```
    repeat
        <dãy câu lệnh>;
    until <điều kiện>;
```

trong đó *điều kiện* là biểu thức quan hệ hoặc lôgic.

Việc thực hiện lệnh *repeat-until* được thể hiện trên sơ đồ sau.



Hình P.8. *Sơ đồ thực hiện lệnh repeat-until*

Ví dụ 1

Xét đoạn chương trình:

```
repeat
   write('HAY NHAP SO NGUYEN N: ');
   readln(N)
until (N>0) and (N <= 100);
```

Đoạn chương trình này dùng để nhập số nguyên N thoả mãn điều kiện $0 < N \leq 100$. Nếu giá trị nguyên đưa vào nằm ngoài khoảng này thì chương trình sẽ yêu cầu nhập lại cho đến khi giá trị nhập vào thoả mãn điều kiện.

Ví dụ 2

Tính gần đúng $\sqrt{A}$ ($A > 0$) chính xác đến chữ số thập phân thứ năm theo phương pháp lặp Niu-tơn.

Theo phương pháp này, ta cần tính dãy $X_0, X_1, X_2,...$, theo công thức:

$X_0 = 1$

…

$$X_{n+1} = \frac{1}{2}(X_n + \frac{A}{X_n}), \qquad n = 0, 1, 2, ... \qquad (*)$$

Quá trình lặp kết thúc khi $|X_{n+1} - X_n| < 0{,}00001$. Giá trị của X_{n+1} được lấy làm giá trị gần đúng của $\sqrt{A}$.

Input: Giá trị thực A, nhập vào từ bàn phím.

Output: Đưa ra màn hình giá trị gần đúng của $\sqrt{A}$.

Ta chỉ cần dùng hai biến thực trung gian *XNEW* và *XOLD* để chứa tương ứng các giá trị X_{n+1} và X_n.

Thuật toán được mô tả như sau:

Bước 1. Nhập A;

Bước 2. $XNEW \leftarrow 1$ {giá trị X_0};

Bước 3. $XOLD \leftarrow XNEW$ {chuyển giá trị mới thành giá trị cũ};

Bước 4. $XNEW \leftarrow \frac{1}{2}(XOLD + \frac{A}{XOLD})$ {Tính lại giá trị mới theo (*)};

Bước 5. Nếu $|XNEW - XOLD| \geq 0{,}00001$ thì quay về bước 3;

Bước 6. Đưa giá trị *XNEW* ra màn hình, rồi kết thúc.

Chương trình thể hiện thuật toán trên như sau:

```
program CAN_BAC_2;
uses  crt;
var  A, XNEW, XOLD: real;
begin
   clrscr;
   repeat
        write(' A = '); readln(A);     {Buoc 1}
   until A > 0;
   XNEW:= 1;   {Buoc 2}
   repeat
        XOLD:= XNEW;                   {Buoc 3}
        XNEW:= (XOLD + A/XOLD)/2;      {Buoc 4}
   until ABS(XNEW - XOLD) < 0.00001;   {Buoc 5}
   writeln('Can bac 2 cua ',A:10:5,' la ',XNEW:10:5);{Buoc 6}
   readln
end.
```

6. Câu lệnh with

Trong chương trình Xep_loai ở §13 ta thấy sau câu lệnh *for-do* mỗi khi truy cập đến một trường của biến bản ghi *Lop*[*i*], ta đều phải viết:

```
Lop[i].<tên trường>
```

Như vậy, cần xử lí bao nhiêu trường thì phải viết kèm bấy nhiêu lần *Lop*[*i*]. Pascal cung cấp lệnh *with* để tham chiếu đến các trường của biến bản ghi một cách ngắn gọn như sau:

```
with   <tên biến bản ghi> do
       <câu lệnh>;
```

Trường (không kèm tên *biến bản ghi*) trong *câu lệnh* được hiểu là trường của *biến bản ghi* đó.

Ví dụ

Ta có thể viết lại chương trình Xep_loai ở §13 như sau:

```
program Xep_loai;
uses crt;
const Max = 45;
type HocSinh = record
                    HoTen: string[30];
                    NgaySinh: string[10];
                    DiaChi: string[50];
                    Toan,Van: real;
                    XepLoai: char;
                end;
var    Lop: array[1..Max] of HocSinh;
       N,i: byte;
begin
   clrscr;
   write('So luong hoc sinh trong lop N = ');
   readln(N);
   for i:=1 to N do
      with lop[i] do
      begin
       writeln('Cac so lieu ve hoc sinh thu ',i,': ');
       write('Ho va ten: '); readln(HoTen);
       write('Ngay sinh: '); readln(NgaySinh);
       write('Dia chi: '); readln(DiaChi);
       write('Diem Toan: '); readln(Toan);
       write('Diem Van: ');  readln(Van);
       if Toan+Van >=18 then XepLoai:= 'A';
       if (Toan+Van>=14) and (Toan+Van<18) then XepLoai:='B';
       if (Toan+Van>=10) and (Toan+Van<14) then XepLoai:='C';
       if Toan+Van<10 then XepLoai:= 'D';
      end;
   clrscr;
   writeln('Danh sach xep loai hoc sinh trong lop:');
   for i:= 1 to N do
        writeln(lop[i].HoTen:30,' -Xep loai:',lop[i].XepLoai);
   readln
end.
```

7. Một số thông báo lỗi

Lỗi biên dịch (Compiler Error)

Mã lỗi	Thông báo lỗi	Ý nghĩa
1	Out of memory	Thiếu bộ nhớ
2	Identifier expected	Phải là tên
3	Unknown identifier	Tên gọi chưa được mô tả
4	Duplicate identifier	Khai báo lặp một tên gọi
5	Syntax error	Lỗi cú pháp
6	Error in real constant	Lỗi khi viết hằng số thực
7	Error in integer constant	Lỗi khi viết hằng số nguyên
8	String constant exceeds line	Hằng xâu vượt quá một dòng
9	Too many nested files	Quá nhiều tệp lồng nhau
10	Unexpected end of file	Không tìm thấy kết thúc tệp
11	Line too long	Dòng quá dài
12	Type identifier expected	Phải là tên kiểu
13	Too many open files	Quá nhiều tệp được mở
14	Invalid file name	Tên tệp không hợp lệ
15	File not found	Không tìm thấy tệp
16	Disk full	Đĩa đầy
18	Too many files	Quá nhiều tệp
20	Variable identifier expected	Phải có tên biến
21	Error in type	Lỗi kiểu
25	Invalid string length	Độ dài xâu không hợp lệ
26	Type mismatch	Sai kiểu (kiểu không thích hợp)
30	Integer constant expected	Phải là hằng nguyên
31	Constant expected	Phải là hằng
32	Integer or real constant expected	Phải là hằng nguyên hoặc hằng thực
34	Invalid function result type	Kiểu kết quả của hàm không hợp lệ
36	BEGIN expected	Thiếu BEGIN
37	END expected	Thiếu END
38	Integer expression expected	Phải là biểu thức nguyên
39	Ordinal expression expected	Phải là biểu thức có thứ tự
40	Boolean expression expected	Phải là biểu thức lôgic
41	Operand types do not match	Kiểu của toán hạng không phù hợp
42	Error in expression	Lỗi trong biểu thức

43	Illegal assignment	Gán không hợp lệ
44	Field identifier expected	Phải là tên trường
50	DO expected	Thiếu DO
54	OF expected	Thiếu OF
57	THEN expected	Thiếu THEN
58	TO or DOWNTO expected	Thiếu TO hoặc DOWNTO
60	Too many procedures	Quá nhiều thủ tục
63	Invalid file type	Kiểu tệp không hợp lệ
64	Cannot read or write variables of this type	Không thể đọc/ghi biến thuộc kiểu dữ liệu này
66	String variable expected	Phải là biến xâu
67	String expression expected	Phải là biểu thức xâu
74	Constant and case types don't match	Hằng và các kiểu của case không phù hợp
75	Record or object variable expected	Phải là biến kiểu bản ghi hoặc đối tượng
76	Constant out of range	Hằng vượt ra ngoài giới hạn
77	File variable expected	Phải là biến tệp
79	Integer or real expression expected	Phải là biểu thức nguyên hoặc thực
85	";" expected	Thiếu dấu ";"
86	":" expected	Thiếu dấu ":"
87	"," expected	Thiếu dấu ","
88	"(" expected	Thiếu dấu "("
89	")" expected	Thiếu dấu ")"
90	"=" expected	Thiếu dấu "="
91	":=" expected	Thiếu dấu ":="
92	"[" or "(." expected	Thiếu dấu "[" hoặc "(."
93	"]" or ".)" expected	Thiếu dấu "]" hoặc ".)"
94	"." expected	Thiếu dấu "."
95	".." expected	Thiếu dấu ".."
96	Too many variables	Quá nhiều biến
97	Invalid FOR control variable	Biến điều khiển FOR không hợp lệ
98	Integer variable expected	Phải là một biến nguyên
100	String length mismatch	Độ dài xâu không phù hợp
101	Invalid ordering of fields	Thứ tự các trường không hợp lệ
102	String constant expected	Phải là hằng xâu
103	Integer or real variable expected	Phải là biến nguyên hoặc thực
104	Ordinal variable expected	Phải là biến có thứ tự

106	Character expression expected	Phải là biểu thức kí tự
113	Error in statement	Lỗi trong câu lệnh
117	Target address not found	Không tìm thấy địa chỉ đích
124	Statement part too large	Thành phần câu lệnh quá lớn
127	Too many conditional symbols	Quá nhiều kí hiệu điều kiện
132	Critical disk error	Lỗi đĩa trầm trọng
139	Cannot access this symbol	Không thể truy cập kí hiệu này
140	Invalid floating-point operation	Phép toán dấu phẩy động không hợp lệ
146	File access denied	Không thể truy cập tệp

Lỗi sinh ra trong quá trình chạy chương trình (Runtime Error)

Mã lỗi	Thông báo lỗi	Ý nghĩa
2	File not found	Không tìm thấy tệp
3	Path not found	Không tìm thấy đường dẫn
100	Disk read error	Lỗi khi đọc đĩa
101	Disk write error	Lỗi khi ghi đĩa
102	File not assigned	Tệp chưa được chỉ định
103	File not open	Tệp chưa được mở
104	File not open for input	Tệp chưa được mở để nhập dữ liệu
105	File not open for output	Tệp chưa được mở để xuất dữ liệu
106	Invalid numeric format	Định dạng số không hợp lệ
150	Disk is write-protected	Đĩa đang được bảo vệ chống ghi
152	Drive not ready	Thiết bị chưa sẵn sàng
156	Disk seek error	Lỗi tìm kiếm trên đĩa
158	Sector Not Found	Không tìm thấy sector được yêu cầu
162	Hardware failure	Lỗi phần cứng
200	Division by zero	Lỗi chia cho số 0
201	Range check error	Vượt phạm vi khai báo
202	Stack overflow error	Lỗi tràn ngăn xếp
203	Heap overflow error	Lỗi tràn vùng nhớ Heap
205	Floating-point overflow	Số dấu phẩy động quá lớn
207	Invalid floating-point operation	Phép toán với số dấu phẩy động không hợp lệ

PHỤ LỤC C

Câu lệnh rẽ nhánh và lặp trong C++

a) Câu lệnh rẽ nhánh

- *Dạng khuyết*

Cú pháp:

if (*điều kiện*) <*câu lệnh*>;

trong đó:

– *Điều kiện* là biểu thức nhận giá trị lôgic được đặt trong cặp ngoặc tròn (và);

– *Câu lệnh* là câu lệnh đơn hoặc ghép.

Khi *điều kiện* nhận giá trị *đúng* thì *câu lệnh* được thực hiện.

Ví dụ. Đoạn chương trình sau đây kiểm tra nếu giá trị biến *x* bằng 100 thì đưa ra thông báo "*x bang* 100":

```
if       (x == 100)
    cout << " x bang 100 ";
```

- *Dạng đủ*

Cú pháp:

if (*điều kiện*) <*câu lệnh 1*>; **else** <*câu lệnh 2*>;

Trong câu lệnh này, khi *điều kiện* nhận giá trị *đúng* thì *câu lệnh 1* được thực hiện, ngược lại thì *câu lệnh 2* được thực hiện.

Ví dụ. Đoạn chương trình sau đây kiểm tra nếu giá trị biến x bằng 100 thì đưa ra màn hình thông báo "*x bang* 100", ngược lại thì thông báo "*x khong bang* 100":

```
if (x == 100)
   cout << " x bang 100 ";
   else cout << " x khong bang 100 ";
```

b) Câu lệnh lặp

- *Câu lệnh lặp* ***while***

Cú pháp:

while (*điều kiện*) <*câu lệnh*>;

Trong câu lệnh này, *điều kiện* là biểu thức nhận giá trị lôgic. Trong khi *điều kiện* còn nhận giá trị đúng thì còn thực hiện *câu lệnh*.

Ví dụ. Nhập từ bàn phím một số tự nhiên *n* (*n* < 20), đưa ra màn hình các số nguyên dương giảm dần từ *n* đến 1.

```
#include <iostream.h>
int main()
{
  int n;
  cout << "Nhap so nguyen duong n (n<20): ";
  cin >> n;
  while (n>0)
       cout << n-- << ", ";
  return 0;
}
```

Trong đó `cout << n--` hiển thị giá trị hiện thời của *n* sau đó giảm giá trị *n* một đơn vị. Vòng lặp được kết thúc khi *điều kiện* nhận giá trị *sai*. Trong ví dụ trên, giá trị của *n* được giảm nhờ toán tử "`--`" nên vòng lặp được kết thúc khi *n* = 0.

- *Câu lệnh lặp* ***do***

Cú pháp:

do <*câu lệnh*> **while** (*điều kiện*);

Sau khi thực hiện *câu lệnh*, vòng lặp sẽ kiểm tra điều kiện, nếu *điều kiện* còn nhận giá trị đúng thì *câu lệnh* còn được thực hiện. *Điều kiện* được tính toán sau khi *câu lệnh* được thực hiện, vì vậy *câu lệnh* sẽ được thực hiện ít nhất một lần.

Ví dụ. Đưa ra màn hình số nguyên nào nhập vào từ bàn phím cho đến khi số được nhập là 0.

```
#include <iostream.h>
int main()
{
  long n;
  do {
   cout << "Nhap vao mot so nguyen(nhap so 0 de ket thuc): ";
   cin >> n;
   cout << "So da nhap la: " << n << "\n";
     }
   while (n != 0);
  return 0;
 }
```

- *Câu lệnh lặp* ***for***

Cú pháp:

for (*khởi tạo; điều kiện; thay đổi*) <*câu lệnh*>;

Vòng lặp *for* thực hiện lặp lại *câu lệnh* chừng nào *điều kiện* còn nhận giá trị *đúng* như trong vòng lặp *while*. Nhưng trong *for* còn chứa khả năng *khởi tạo* và khả năng làm *thay đổi* giá trị các biến điều khiển vòng lặp.

Hoạt động của vòng *for* như sau:

Bước 1. Đầu tiên *khởi tạo* được thực hiện thường để đặt một giá trị ban đầu cho các biến điều khiển, điều này chỉ được thực hiện một lần.

Bước 2. *Điều kiện* được kiểm tra, nếu nó nhận giá trị đúng thì *câu lệnh* được thực hiện, ngược lại thì tới bước 4.

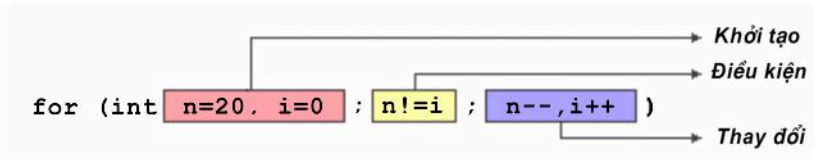
Bước 3. *Thay đổi* được thực hiện để thay đổi giá trị biến điều khiển, quay về bước 2.

Bước 4. Kết thúc vòng lặp.

Ví dụ. Đưa ra màn hình các cặp số tự nhiên khác nhau có tổng bằng 20.

```
#include <iostream.h>
int main()
{
  for (int n=20,i=0; n!=i; n--,i++)
    cout << n << ", " << i <<'\n';
  return 0;
}
```

Giải thích:



Ban đầu có cặp số (n = 20; i = 0) được đưa ra màn hình. Để tạo các cặp số khác, cho n giảm dần từng đơn vị, đồng thời i tăng dần từng đơn vị (để tổng của chúng vẫn bằng 20). Vòng lặp kết thúc khi gặp cặp số (n = 10; i = 10).

MỤC LỤC

Trang

CHƯƠNG I . MỘT SỐ KHÁI NIỆM VỀ LẬP TRÌNH
VÀ NGÔN NGỮ LẬP TRÌNH 3
§1. Khái niệm lập trình và ngôn ngữ lập trình 4
Bài đọc thêm 1. Bạn biết gì về các ngôn ngữ lập trình? 6
§2. Các thành phần của ngôn ngữ lập trình 9
Bài đọc thêm 2. Ngôn ngữ Pascal 14
CHƯƠNG II. CHƯƠNG TRÌNH ĐƠN GIẢN 17
§3. Cấu trúc chương trình 18
§4. Một số kiểu dữ liệu chuẩn 21
§5. Khai báo biến 22
§6. Phép toán, biểu thức, câu lệnh gán 24
§7. Các thủ tục chuẩn vào/ra đơn giản 29
§8. Soạn thảo, dịch, thực hiện và hiệu chỉnh chương trình 32
Bài tập và thực hành 1 34
CHƯƠNG III. CẤU TRÚC RẼ NHÁNH VÀ LẶP 37
§9. Cấu trúc rẽ nhánh 38
§10. Cấu trúc lặp 42
Bài tập và thực hành 2 49
CHƯƠNG IV. KIỂU DỮ LIỆU CÓ CẤU TRÚC 52
§11. Kiểu mảng 53
Bài tập và thực hành 3 63
Bài tập và thực hành 4 65
§12. Kiểu xâu 68
Bài tập và thực hành 5 73
§13. Kiểu bản ghi 74

CHƯƠNG V. TỆP VÀ THAO TÁC VỚI TỆP 81
§14. Kiểu dữ liệu tệp 82
§15. Thao tác với tệp 83
§16. Ví dụ làm việc với tệp 87
CHƯƠNG VI. CHƯƠNG TRÌNH CON VÀ LẬP TRÌNH CÓ CẤU TRÚC 90
§17. Chương trình con và phân loại 91
§18. Ví dụ về cách viết và sử dụng chương trình con 96
Bài tập và thực hành 6 103
Bài tập và thực hành 7 105
Bài đọc thêm 3. Ai là lập trình viên đầu tiên? 109
§19. Thư viện chương trình con chuẩn 110
Bài tập và thực hành 8 115
Bài đọc thêm 4. Âm thanh 118
PHỤ LỤC A
1. Một số phép toán thường dùng 121
2. Giá trị phép toán lôgic 121
PHỤ LỤC B
1. Môi trường Turbo Pascal 122
2. Một số tên dành riêng 128
3. Một số kiểu dữ liệu chuẩn 129
4. Một số thủ tục và hàm chuẩn 129
5. Câu lệnh rẽ nhánh và lặp 131
6. Câu lệnh with 134
7. Một số thông báo lỗi 136
PHỤ LỤC C
Câu lệnh rẽ nhánh và lặp trong C++ 139

Chịu trách nhiệm xuất bản : Chủ tịch Hội đồng Thành viên **NGUYỄN ĐỨC THÁI**
Tổng Giám đốc **HOÀNG LÊ BÁCH**
Chịu trách nhiệm nội dung : Tổng biên tập **PHAN XUÂN THÀNH**

Biên tập lần đầu : **NGUYỄN THỊ THANH XUÂN – PHẠM THỊ THANH NAM**
Biên tập tái bản : **NGUYỄN THỊ NGUYÊN THUÝ**
Biên tập kĩ thuật : **HOÀNG VIỆT HÙNG - TRẦN THANH HẰNG**
Trình bày bìa : **LƯƠNG QUỐC HIỆP**
Sửa bản in : **DƯƠNG VŨ KHÁNH THUẬN**
Chế bản : **CÔNG TY CP DỊCH VỤ XUẤT BẢN GIÁO DỤC HÀ NỘI**

TIN HỌC 11

Mã số : CH118T0

In..... cuốn (QĐ in số : .....), khổ 17 × 24 cm.
Đơn vị in : ..... địa chỉ .....
Cơ sở in : ..... địa chỉ .....
Số ĐKXB : 01 - 2020/CXBIPH/608 - 869/GD
Số QĐXB : ... / QĐ-GD ngày ... tháng ... năm ....
In xong và nộp lưu chiểu tháng ... năm ...
Mã số ISBN : 978-604-0-18887-8