Trường Đại học Bách Khoa TP HCM Khoa Khoa học & Kỹ thuật Máy tính

MP

Micro Ngôn ngữ Pascal

Tác giả Tiến sĩ Nguyễn Hua Phụng

Tháng 6 năm 2017

Nội dung

MP

phiên bản 1.2

1 Giới thiệu

MP (Mini Pascal) là một ngôn ngữ bao gồm một tập hợp con của Pascal cộng với một số tính năng ngôn ngữ Java. Các tính năng Pascal của ngôn ngữ này là (các chi tiết sẽ được thảo luận sau): một vài kiểu nguyên thủy, mảng một chiều, cấu trúc điều khiển, biểu thức, câu lệnh ghép (tức là khối), hàm và thủ tục.

Các tính năng Java của ngôn ngữ này như sau:

1. MP có nhiều bài tập, mượn từ Java.
2. Trong MP, như Java, toán hạng của toán tử được đảm bảo được đánh giá theo thứ tự đánh giá cụ thể, đặc biệt, từ trái sang phải. Trong Pascal, thứ tự đánh giá không được chỉ định. Trong trường hợp của *f () + g ()*, ví dụ, MP dictates rằng *e* luôn đánh giá trước *g* .

Vì lý do đơn giản:

1. Chỉ có một mảng kích thước trong MP.

```
var i: mảng [1. . 2, 3. . 4] trong số số nguyên ;          // LỖI
var i: mảng [1. . 5] trong số số nguyên ;          // CHÍNH XÁC
```

Thông thường, **chuỗi '\ n' phải được sử dụng làm ký tự dòng mới trong MP** .

2 Chương trình Kết cấu

MP không hỗ trợ biên dịch riêng biệt nên tất cả các khai báo (biến và hàm) phải được cư trú trong một tệp duy nhất.

Một chương trình MP bao gồm nhiều khai báo có thể là một khai báo biến hoặc chức năng hoặc thủ tục.

2.1 Biến tờ khai:

Một khai báo biến bắt đầu với từ khóa **var** và sau đó một danh sách các khai báo mà mỗi bắt đầu với một danh sách phân tách bằng dấu phẩy, dấu hai chấm (:), một kiểu và kết thúc bằng dấu chấm phẩy.
Ví dụ,

```
var a, b, c: số nguyên ;
        d: mảng [1. . 5] của số nguyên ; e, f: thật ;
```

2.2 Khai báo hàm :

Trong MP, khai báo hàm xác định tên của hàm, kiểu giá trị trả về và số và kiểu của các đối số phải được cung cấp trong một cuộc gọi đến hàm như tốt như các thân hình của các chức năng. A chức năng Là khai báo như sau:
function *<function-name> '(' <parameter-list> ')' ':' <return type> ';' <biến decla- ration> <compound-statement>*
Ở đâu

- ***chức năng*** Là một từ khóa
- *<function-name>* là số nhận dạng được sử dụng để thể hiện tên của chức năng.

  <parameter-list> là zero hoặc nhiều *<parameter-khai>* 's Ly bởi ';'. A
  <parameter-declaration> được khai báo như sau:
  <danh sách số nhận dạng được phân tách bằng dấu phẩy> ':' <type>

- *<type>* là hàm trả về kiểu.

  <khai báo biến> được mô tả trong phần trước. Tuyên bố biến có thể là làm ngơ.

- *<compound-statement>* được mô tả trong Phần 6.8 cho thí dụ,

hàm foo (a, b: số nguyên; c: thực): mảng [1. . 2] của số nguyên;

var x, y: real; được g trong
. . .
kết thúc

2.3 Thủ tục tờ khai:

Khai báo thủ tục giống như một hàm ngoại trừ ***thủ tục*** từ khóa được sử dụng thay cho ***hàm*** từ khóa và không có dấu hai chấm và kiểu trả về sau khai báo tham số. phần.

Ví dụ,

p ro ce d bạn lại f o o (một , b : tôi n t e g e r ; c : r e một l) ; var x,
y: real ;
được g trong
. . .
kết thúc

MP không hỗ trợ quá tải chức năng / thủ tục. Do đó, một hàm phải được xác định chính xác một lần.
MP không hỗ trợ hàm / thủ tục lồng nhau như trong Pascal. Ví dụ: khai báo tiếp theo không hợp lệ trong MP

hàm foo (i: số nguyên): thực;
p ro ce d bạn lại đứa trẻ _ o f _ f o o (r e một l f) là g trong
. . .
kết thúc
được g vào cuối *// ERROR*

3 Lexical Speciftcation

Phần này mô tả bộ ký tự, quy ước nhận xét và bộ mã thông báo bằng ngôn ngữ.

3.1 Tính cách Bộ

Một chương trình MP là một chuỗi ký tự từ bộ ký tự ASCII. Trống, tab, formfeed (ví dụ, ASCII FF), vận chuyển trở lại (ví dụ, CR ASCII) và dòng mới (tức là, ASCII LF) là các *ký tự khoảng trắng* . Một dòng là một dãy các ký tự kết thúc bằng một LF. Định nghĩa của các dòng này có thể được sử dụng để xác định số dòng được tạo ra bởi trình biên dịch M.

3.2 Bình luận

Có ba loại bình luận:

Nhận xét khối truyền thống:
(Đây là*
*một bình luận khối *)*
Tất cả văn bản từ (* đến *) đều bị bỏ qua.

Nhận xét chặn:
{ Đây là một bình luận khối }
Tất cả văn bản từ { đến } đều bị bỏ qua.

Nhận xét dòng:
// Đây là nhận xét dòng
Tất cả văn bản từ // đến cuối dòng đều bị bỏ qua.
Như được thiết kế trong C, C ++ và Java, các quy tắc sau cùng được thực thi:

- - Nhận xét không làm tổ.
- - (*, *), { và } có Không đặc biệt Ý nghĩa trong bình luận cái đó bắt đầu với //.
- - // không có ý nghĩa đặc biệt trong các nhận xét bắt đầu bằng (* hoặc {.

3,3 Mã thông báo Bộ

Trong một chương trình MP, có năm loại mã thông báo: số nhận dạng, từ khóa, toán tử, dấu tách và chữ.

Số nhận dạng: Mã định danh là chuỗi ký tự, chữ số và số điểm không giới hạn, số đầu tiên phải là một chữ cái hoặc dấu gạch dưới. MP **phân biệt chữ hoa chữ thường** , có nghĩa là *abc* và *Abc* giống nhau.

Từ khóa: Các chuỗi ký tự sau được đặt dưới dạng *từ khóa* và không thể được sử dụng làm số nhận dạng:
phá vỡ tiế p tục để downto làm nế u sau đó khác trở lại trong khi bắ t đầ u kế t thúc chức năng thủ tục var đúng mảng giả của chuỗi số nguyên boolean không và hoặc div mod
Một số từ khóa được sử dụng làm toán tử hoặc chữ. Những từ khóa này cũng **phân biệt chữ hoa chữ thường** .

- Toán tử: 15 *toán tử sau* trong MP:

Nhà điề u hành	Ý nghĩa	Nhà điề u hành	Ý nghĩa
+	Thêm vào	- -	Trừ hoặc phủ định
*	Phép nhân	/	Bộ phận
không phải	Hợp lý KHÔNG	mod	Mô đun
hoặc là	Hợp lý HOẶC	và	Logic và
<>	Không công bằng	=	Công bằng
<	Ít hơn	>	Lớn hơn
<=	Nhỏ hơn hoặc bằng	> =	Lớn hơn hoặc bằng
div	Phân chia số nguyên		

3,4 Dấu phân tách

Các ký tự sau là **dấ u phân cách** : **dấ u** ngoặc vuông trái ('[') dấu ngoặc vuông phải (']'), dấu hai chấm (':'), dấu ngoặc trái ('('), dấu ngoặc vuông (')'), dấu chấm phẩy (';'), dấu chấm kép ('..') và dấu phẩy (',').

3,5 Chữ viết

Một **chữ** là một biểu diễn nguồn của một giá trị của một kiểu số nguyên, kiểu thực, kiểu boolean hoặc kiểu chuỗi.

Một *số nguyên nghĩa đen* **luôn** được **thể hiện bằ ng chữ số thập phân (cơ sở 10),** bao gồm một quence se- của ít nhất một chữ số. Một chữ số nguyên là loại **số nguyên** .

Một *dấu chấm đen* có các phần sau đây: một phần toàn bộ số, một dấu thập phân (đại diện bởi một nhân vật kỳ ASCII), một phần phân đoạn và một số mũ. Số mũ, nếu có, được chỉ định bởi các **e** thư ASCII hoặc **E** tiếp theo là một tùy chọn ký (-) số nguyên. Có ít nhất một chữ số, trong cả số hoặc phần phân số và dấu thập phân hoặc số mũ được yêu cầu. Tất cả các bộ phận khác là tùy chọn. A điểm nổi theo nghĩa đen Là của kiểu **thật** .
Ví dụ: Sau đây là các chữ nổi hợp lệ:
1.2 1. .1 1e2 1.2E-2 1.2e-2 .1E2 9.0 12e8 0.33E-3 128e-42
Sau đây **không** được coi là chữ nổi: e-12 (không có chữ số trước 'e') 143e
(không có chữ số sau 'e')

Chữ *boolean* có hai giá trị, được biểu diễn bằng các chữ **cái đúng** và **sai** , được tạo thành từ các chữ cái ASCII.

Một *chuỗi ký tự* bao gồm số không hoặc nhiều ký tự được đặt trong dấu ngoặc kép " '. Dấu ngoặc kép không phải là một phần của chuỗi, nhưng phân phối để phân tách nó.
Nó là một lỗi biên dịch thời gian cho một backspace, newline, formfeed, vận chuyển trở lại, tab, báo giá duy nhất, dấu ngoặc kép hoặc dấu gạch chéo ngược xuất hiện sau khi mở " 'và trước

các chuỗi thoát sau được sử dụng thay thế:

\ b	backspace
\ f	thức ăn dạng
\ r	xe trở về
\ n	dòng mới
\ t	ngang chuyển hướng
\ '	Độc thân Trích dẫn
\ "	gấp đôi Trích dẫn
\\	dấu chéo ngược Một

chuỗi ký tự là loại **chuỗi** .

4 Các loại và Giá trị

Các loại của tất cả các biến và biểu thức trong MP phải được biết tại thời gian biên dịch. Các loại giới hạn các giá trị mà một biến có thể giữ (ví dụ, một số nhận dạng x có kiểu số nguyên không thể giữ giá trị đúng ...), các giá trị mà một biểu thức có thể tạo ra, và các hoạt động được hỗ trợ trên các giá trị đó (ví dụ, chúng tôi không thể ứng dụng một thêm nhà điều hành đến 2 boolean giá trị. . .).
Các loại MP được chia thành hai loại:

- Nguyên thủy loại: **boolean, sô´ nguyên, thực, chuỗi** .
- Loại hợp chất: **mảng** .

4.1 Các boolean Kiểu và Giá trị

Kiểu **boolean** đại diện cho một số lượng hợp lý với hai giá trị có thể: *true* và *false* . Các tiếp theo toán tử có thể hành động trên boolean giá trị:
không, và, và sau đó, hoặc, hoặc người nào khác
Một *biểu thức boolean* là một biểu thức để đánh giá *đúng* hay *sai*. Biểu thức Boolean xác định luồng điều khiển trong **if** , **for** và **while** . Chỉ các biểu thức boolean mới có thể được sử dụng trong các câu lệnh luồng điều khiển này.
Trong khi toán tử đầu tiên (không) là unary, những người khác là nhị phân. Các toán tử *và* và *hoặc* được đánh giá bình thường có nghĩa là toán hạng của chúng được tính toán trước các toán tử này. Trong khi đó, các nhà khai thác *và sau đó* và *hoặc người nào khác* được đánh giá ngắn mạch có nghĩa là các trái toán hạng Là đánh giá trước tiên và các đúng một Là đánh giá chỉ có khi nào cần thiết.

4.2 Các số nguyên Kiểu và Giá trị

Các giá trị của **sô´ nguyên** là các **sô´ nguyên** được ký 32 bit trong các phạm vi sau:
-2147483648. . . 2147483647
Các toán tử sau đây có thể hoạt động trên các giá trị số nguyên:
+ - * div mod <<=>> = <> = /
Nặm toán tử đầu tiên luôn tạo ra một giá trị kiểu **sô´ nguyên** . Sáu toán tử tiếp theo luôn dẫn đến một giá trị kiểu **boolean** . Toán tử cuối cùng (/) sẽ dẫn đến giá trị của kiểu **thực** khi nào cả hai toán hạng là trong kiểu **sô´ nguyên** .
Ở đây, - đại diện cho cả phép trừ nhị phân và toán tử phủ định đơn nhất.

4.3 Các thực Kiểu và Giá trị

Giá trị thực là số có độ chính xác 32 bit. Các giá trị chính xác của loại này phụ thuộc vào việc triển khai thực hiện. Các toán tử sau đây có thể hành động trên các giá trị dấu phẩy động:

- Toán tử số học nhị phân +, -, * và /, dẫn đến giá trị kiểu **thật** .
- Các đoàn kết sự phủ nhận toán tử -, cái nào các kết quả trong một giá trị của kiểu **thật** .

Các toán tử quan hệ =, <>, <, <=,> và> =, dẫn đến giá trị kiểu **boolean** .

4.4 Các chuỗi Kiểu và Giá trị

Các chuỗi chỉ có thể được sử dụng trong một nhiệm vụ hoặc được truyền như một tham số của một lời gọi hàm. Ví dụ, một chuỗi có thể được chuyển thành một tham số cho hàm *putString ()* hoặc *putStringLn () được tích hợp sẵn* như được mô tả trong Phần 7.

4,5 Mảng Các loại và Của chúng Giá trị

MP chỉ hỗ trợ **mảng một chiê`u** . Ban đầu, các mảng trong Pascal hỗ trợ các tính năng sau:

Giới hạn dưới và giới hạn trên của chỉ mục phải được cung cấp trong khai báo mảng.

- Một chỉ số có thể là bất kỳ biểu thức số nguyên nào, tức là, bất kỳ biểu thức nào của kiểu **sô´ nguyên** . Tuy nhiên, cho sự đơn giản mục đích, một chiều mảng trong MP là hơn hạn chế :

Loại phần tử của một mảng chỉ có thể là một kiểu nguyên thủy như **boolean** , **integer** , **real** hoặc **string** .

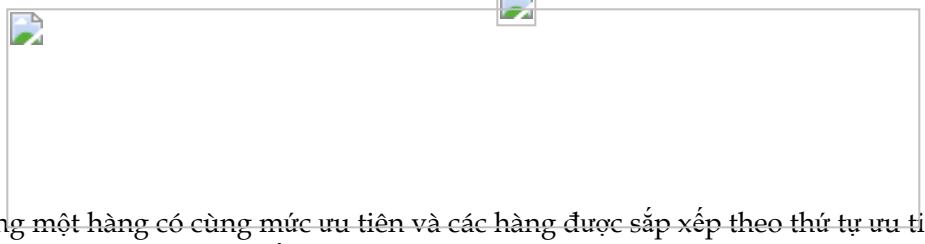
Một biến mảng chính nó (không có dấu ngoặc vuông liên quan []) chỉ có thể được sử dụng như một thực tế tham số đến vượt qua đến từ một chức năng hoặc là thủ tục.

5 Biểu thức

Một biểu thức là một sự kết hợp hữu hạn của toán hạng và toán tử. Toán hạng của một biến thể có thể là một chữ, một định danh, một phần tử của một mảng hoặc một cuộc gọi hàm.

5.1 Ưu tiên và Khả năng tương tác

Các quy tắc ưu tiên và kết hợp của các toán tử được hiển thị như sau:



Các toán tử trên cùng một hàng có cùng mức ưu tiên và các hàng được sắp xếp theo thứ tự ưu tiên giảm dần. Biểu thức nằm trong '(' và ')' có mức ưu tiên cao nhất.

5,2 Kiểu Coercions

Trong MP, như C và Java, các biểu thức chế độ hỗn hợp có các toán hạng có các kiểu khác nhau được cho phép.

Các toán hạng của các toán tử sau:

+ - * / <<=>> = = <>

có thể có **sô´ nguyên** hoặc **sô´ thực** . Nếu một toán hạng là có **thật** , trình biên dịch sẽ ngầm chuyển đổi cái kia sang **thực** . Do đó, nếu ít nhất một trong các toán hạng của toán tử nhị phân ở trên là kiểu **thực** , thì thao tác là một phép toán dấu phẩy động.

Các quy tắc cưỡng chế kiểu sau cho phép gán ngầm hoặc rõ ràng được cho phép:

Nếu loại LHS là có **thật** , biểu thức trong RHS phải có loại **sô´ nguyên** hoặc lỗi thời gian **thực** hoặc biên dịch xảy ra.

5.3 Mục lục Biểu hiện

Một **nhà điê`u hành chỉ sô´** được sử dụng để tham khảo hoặc trích xuất các thành phần được chọn của một mảng. Nó phải mất các tiếp theo hình thức:

<expression> '[' expression ']'

Kiểu của *<expression>* đầu tiên phải là một kiểu mảng. Biểu thức thứ hai, tức là biểu thức giữa '[' và ']', phải là kiểu số nguyên. Toán tử index trả về phần tử tương ứng của mảng.

Ví dụ,

foo (2) [3 + x]: = a [b [2]] +3;

Việc gán ở trên là hợp lệ nếu các biến a, b và kiểu trả về của hàm foo nằm trong một kiểu mảng, x là kiểu **sô´ nguyên** và kiểu phần tử của mảng b là **sô´ nguyên** .

5,4 Yêu cầu Biểu hiện

Một biểu thức gọi là một cuộc gọi hàm bắt đầu với một định danh theo sau là "(" và ")". Một danh sách các biểu thức có thể phân tách bằng dấu phẩy có thể xuất hiện giữa "(" và ")" như một danh sách các đối số.

Giống như C, tất cả các đối số (bao gồm mảng) trong MP được chuyển "theo giá trị". Hàm được gọi là giá trị của nó trong các tham số của nó. Do đó, hàm được gọi không thể thay đổi biến trong hàm gọi trong bất kỳ đường.

Khi một hàm được gọi, mỗi tham số của nó được khởi tạo với giá trị của đối số tương ứng được truyền từ hàm người gọi.

Khi một biến mảng được truyền (như một đối số) đến / từ một hàm / thủ tục, thấp hơn giới hạn, giới hạn trên và các loại nguyên tố của các đối số mảng và các thông số chính thức mảng phải giống nhau. Tất cả các thành viên của đối số mảng sẽ được sao chép vào tương ứng các thành viên của các mảng chính thức tham số.

Các quy tắc cưỡng chế kiểu cho phép gán được áp dụng cho tham số truyền trong đó LHS là các tham số chính thức và RHS là các đối số.

Ví dụ,

thủ tục foo (a: **mảng** [1.. 2] **của thực**). . .
thủ tục goo (x: **mảng** [1.. 2] **của thực**);
var
y: **mảng** [2 . . 3] **của thực** ;
z: **mảng** [1 . . 2] **của sô´ nguyên** ;
bă´t đâ`u
foo (x); // CORRECT foo (y); // WRONG f o o (
z) ; //SAI RÒI
kê´t thúc

Các quy tắc cưỡng chế kiểu và ngoại lệ trong việc truyền tham số cũng được áp dụng cho kiểu trả về trong đó LHS là kiểu trả về và RHS là biểu thức trong câu lệnh trả về.

Ví dụ,

f bạn n c t tôi o n f o o ( ) : r e một l ; được
g trong
tôi f (...) sau đó **trả vê`** 2 . 3 ; *//CHÍNH XÁC*
khác trở lại 2; *//CHÍNH XÁC*
kết thúc và,
f bạn n c t tôi o n f o o (b : mảng [1. . 2] o f tôi n t e g e r) : mảng [2 . . 3] trong số thực ;

```
var
    a: mảng [ 2        . .        3] trong số thực ;
được g trong
    if () sau đó trả vê` a ; //CHÍNH XÁC
    khác trở lại b; //SAIRỒI
kết thúc
```

5,5 Đánh giá Gọi món

MP yêu cầu toán hạng bên trái của toán tử nhị phân phải được đánh giá đầu tiên trước khi bất kỳ phần nào của toán hạng bên phải được đánh giá.

Tương tự, trong một cuộc gọi hàm (gọi là một cuộc gọi phương thức trong Java), các tham số thực tế phải được đánh giá từ trái sang đúng.

Mọi toán hạng của toán tử phải được đánh giá trước khi bất kỳ phần nào của thao tác được thực hiện. Hai trường hợp ngoại lệ là các nhà khai thác hợp lý *và sau đó* và *nếu không,* đó vẫn được đánh giá từ trái sang phải, nhưng nó được đảm bảo rằng đánh giá sẽ ngừng ngay sau khi sự thật và sự giả dối Là đã biết. Điều này Là đã biết như các **ngă´n mạch đánh giá** .

6 Báo cáo và kiểm soát lưu lượng

MP hỗ trợ các câu lệnh sau: **chuyển nhượng** , **nê´u** , **cho** , **trong khi** , **ngă´t** , **tiê´p tục** , **trả lại** , **gọi** , **hợp châ´t** và **với** . Tất cả các câu lệnh ngoại trừ **nê´u** , **cho** , văn bản, **hợp châ´t** và một **với** một phải được theo sau bởi một dấu chấm phẩy.

6.1 Nhiệm vụ Tuyên bố

Giống như Java, câu lệnh gán có thể có nhiều mặt bên tay trái nhưng chỉ một bên tay phải. Tuyên bố chuyển nhượng có thể được viết như sau:

lhs $_1$ *: = lhs* $_2$ *: = ... lhs* $_n$ *: = <expression>*

Các cạnh bên tay trái (*lhs* $_1$, ..., *lhs* $_n$) có thể là một biến vô hướng hoặc một biểu thức chỉ mục. Dành cho thí dụ,

```
a: = b [1 0]: = foo () [3]: = x: = 1;
```

Các *<expression>* được tính đầu tiên và giá trị của nó được gán cho *LHS* $_n$. Sau đó, giá trị của *lhs* $_n$ được gán cho *lhs* $_{n-1}$ và cứ thế. Quy tắc cưỡng chế loại được áp dụng cho mỗi nhiệm vụ.

6.2 Nếu Tuyên bố

Có hai loại câu lệnh **if:** if-else và if-không khác. Các if-else được viết như sau:

```
nếu <expression> thì
    <statement1>

khác
    <statement2>
```

trong đó *<expression>* , phải là kiểu **boolean** , được đánh giá đầu tiên. Nếu nó là *sự thật* , *<statement1>* được thực thi. Các *<statement2>* là nếu không thực thi.

Nếu không có khác giống như if-else nhưng không có *else* và *<statement2>* . Trong kiểu câu lệnh if này, nếu *<expression>* là *false* , câu lệnh tiếp theo sẽ được thực hiện.

Giống như C, C ++ và Java, ngôn ngữ MP bị từ cái gọi là lơ lửng-khác. MP giải quyết điều này bằng cách tuyên bố răng người khác phải thuộc về trong cùng nhất nêu.

6,3 Trong khi Tuyên bố

trong khi <expression> làm <statement>

Khi câu lệnh **while** được thực thi, *<expression>* được đánh giá đầu tiên. Trong khi giá trị của nó là *true* , *<statement>* được thực hiện. Vòng lặp dừng khi *<expression>* là *false* .

6,4 Cho Tuyên bố

Câu lệnh **for** được viết như sau:
cho <identifier>: = < expression $_1$ *> (to | downto) < expression* $_2$ *> do <statement>*

trong đó < *expression* $_1$ > được thực thi đầu tiên và giá trị của nó được gán cho *<identifier>* . Nếu từ khóa *để* được sử dụng và giá trị của *<định danh>* là nhỏ hơn hoặc bằng với giá trị của *<biểu thức 2>* thì *<câu>* được thực hiện và sau đó giá trị của *<định danh>* được trong- nếp nhăn một. Sự so sánh giữa *<identifier>* và < *expression* $_2$ > , việc thực thi
<statement> và việc tăng *<identifier>* được lặp lại cho đến khi kết quả của compari trở thành false. Nếu từ khóa *<downto>* được sử dụng, quá trình xảy ra giống nhau, ngoại trừ toán tử quan hệ lớn hơn hoặc bằng và tăng *<identfier>* được thay thế bởi các giảm bớt.

Các *<identifie>* phải là một biến số nguyên địa phương.

6,5 Nghỉ giải lao Tuyên bố

Câu lệnh này phải xuất hiện bên trong một vòng lặp như **cho** hoặc **trong khi** . Khi nó được thực thi, điều khiển sẽ chuyển sang câu lệnh kế bên vòng lặp kèm theo. Tuyên bố này được viết là sau:
phá vỡ ';'

6,6 continue Tuyên bố

Tuyên bố này phải xuất hiện bên trong một vòng lặp như **cho** hoặc **làm trong khi** . Khi nó được thực thi, điều khiển sẽ nhảy lên đến các kết thúc của các thân hình của các vòng lặp. Điều này tuyên bố Là bằng văn bản như

sau:
tiếp tục ';'

6,7 Sự trở lại Tuyên bố

Một tuyên bố **trở lại** nhằm chuyển giao quyền kiểm soát cho người gọi của hàm / thủ tục có chứa nó.

Một câu lệnh **return** không có biểu thức phải được chứa trong một thủ tục trong khi **trở vê`** với biểu hiện phải được trong một chức năng.

6,8 Hợp chất Tuyên bố

Một tuyên bố kết **hợp** bắt đầu với từ khóa **bă´t đâ`u** và kết thúc bằng từ khóa **kê´t thúc** . Giữa hai từ khóa này, có một danh sách các câu lệnh không có giá trị.

6,9 Với tuyên bố

A **với** tuyên bố được viết bằng một hình thức:
với *<list of declaration declaration>* ***do*** *<statement>*

Ví dụ,

```
với một , b : tôi n t e g e r ; c : mảng          [1. . 2 ]          của thực ; làm d =
        c [a] + b ;
```

6,10 Gọi điện tuyên bố

Một câu lệnh gọi bắt đầu bằng một *<identifier>* , là một tên thủ tục, theo sau là một danh sách các biểu thức được phân tách bằng dấu phẩy có dấu ngoặc kép được bao quanh bởi các dấu ngoặc tròn. Ví dụ,

foo (3, a + 1 , m (2));

7 Các hàm dựng sẵn

MP có một số chức năng tích hợp sau:
Hàm getInt (): integer : đọc và trả về một giá trị nguyên từ *thủ tục* nhập chuẩn *putInt (i: integer)* : in giá trị của số nguyên i vào *thủ tục* đầu ra tiêu chuẩn *putIntLn (i: integer)* : giống như putInt ngoại trừ nó cũng in một *hàm* newline *getFloat (): real* : đọc và trả về một giá trị dấu phẩy động từ *thủ tục* nhập chuẩn *putFloat (f: real)* : in giá trị của giá trị thực f vào tiêu chuẩn đầu ra
thủ tục putFloatLn (f: real) : giống như putFloat ngoại trừ việc nó cũng in một *thủ tục* newline *putBool (b: boolean)* : in giá trị của boolean b vào *thủ tục output putBoolLn (b: boolean)* : giống như putBoolLn ngoại trừ nó cũng in một dòng mới

procedure putString (s: string) : in giá trị của chuỗi vào *thủ tục* đầu ra tiêu chuẩn *putStringLn (s: string)* : giống như putStringLn ngoại trừ việc nó cũng in một *thủ tục* dòng mới *putLn ()* : in một dòng mới vào đầu ra tiêu chuẩn

số 8 Phạm vi Quy tắc

Các quy tắc phạm vi điều chỉnh các khai báo (xác định các lần xuất hiện của các số nhận dạng) và cách sử dụng của chúng (ví dụ: các lần nhận dạng được áp dụng).

Phạm vi của một tuyên bố là khu vực của chương trình mà trên đó tờ khai có thể được chuyển đến. Một tuyên bố được cho là nằm trong phạm vi tại một điểm trong chương trình nếu phạm vi của nó bao gồm điểm.

Có ba mức phạm vi: toàn cầu, hàm / thủ tục và khối.

Phạm vi toàn cầu là toàn bộ chương trình được áp dụng cho tất cả các hàm / thủ tục giải mã và khai báo biến bên ngoài các khai báo hàm / thủ tục. Tất cả các hàm / thủ tục dựng sẵn được áp dụng cho phạm vi này

Phạm vi hàm / thủ tục là toàn bộ hàm / thủ tục tương ứng được áp dụng cho các tham số và khai báo biến của nó ngay sau phần tham số.

Một phạm vi khối là câu lệnh trong câu lệnh *with* . Phạm vi được áp dụng cho các khai báo giữa các từ khóa **có** và **làm** .

Có bốn quy tắc bổ sung về giới hạn phạm vi:

1. Tất cả các khai báo trong toàn cầu phạm vi là có hiệu lực trong các toàn thể progran.
2. Tất cả các khai báo trong phạm vi địa phương có hiệu lực từ nơi khai báo đến cuối của nó phạm vi.
3. Không định danh có thể được xác định hơn hơn Một lần trong các tương tự phạm vi. Điều này ngụ ý cái đó **Không** số nhận dạng đại diện cho cả biến toàn cầu và tên hàm cùng một lúc.
4. *Hầu hết các quy tắc lồng nhau đã đóng* : Đối với mỗi lần xuất hiện được áp dụng của một số nhận dạng trong một khối, phải có một khai báo tương ứng, đó là trong khối bao quanh nhỏ nhất có chứa bất kì tờ khai của cái đó số nhận dạng.

Xem xét chương trình MP sau đây:

```
1  var i: sô´ nguyên ;
2  chức năng f (): sô´ nguyên ;
3  bă´t đâ`u
4        r e tu r n 2 0 0 ;
5  kê´t thúc
6  thủ tục chính () ;
```

```
7   var
số 8          chính : số nguyên ;
9   bắt đầu
10          chính : = f () ;
11          putIntLn ( chủ yếu ) ;
12          với
13                  i: số nguyên ;
14                  chủ yếu : số nguyên ;
15                  f: số nguyên ;
16          làm bắt đầu
17        chính : = f : = i : = 1 0 0 ;
18                  putIntLn ( tôi ) ;
19                  putIntLn ( chủ yếu ) ;
20                  putIntLn ( f ) ;
21          kết thúc
22          putIntLn ( chủ yếu ) ;
23  kết thúc
24  var g: real ;
```

Chương trình trên sẽ được biên dịch và in ra các kết quả sau:

200
100
100
100
200

Trong chương trình này, có ba mức phạm vi:

Lưu ý rằng biến *g được* khai báo trong dòng 24 có phạm vi toàn cục mặc dù nó được khai báo ở cuối chương trình.

Biến *chính được* khai báo trong dòng 8 được cho là *ẩn chính* khai báo thủ tục trong dòng 6. Biến *chính được* khai báo trong dòng 14 ẩn *chính* khai báo biến xếp hàng

8. Biến *tôi đã* khai báo trong dòng 13 ẩn biến toàn cục *i* trong dòng 1. Biến *f*

được khai báo trong dòng 15 ẩn khai báo hàm *f* trong dòng 2.

Phạm vi của các khai báo f trong dòng 2, i trong dòng 1 và dòng chính trong dòng 6 không tiếp giáp. Khoảng trống như vậy được gọi là lỗ phạm vi, nơi các khai báo tương ứng được ẩn hoặc ẩn. Như một vấn đề về phong cách, khuyên không nên giới thiệu các biến che giấu tên trong phạm vi bên ngoài. Đây là lý do chính tại sao Java không cho phép khai báo biến từ việc ẩn một khai báo biến khác có cùng tên trong phạm vi bên ngoài. Do đó, chương trình MP ở trên Là một xấu lập trình Phong cách.

9 Chính chức năng

Một thủ tục đặc biệt, tức là thủ tục chính, là một mục nhập của một chương trình MP nơi chương trình bắt đầu:

```
p ro ce du tái chính (); // không có thông số nào là thấp
được g trong
    . . .
kết thúc
```

10 Thay đổi Nhật ký

- Khác nhau từ phiên bản 1.1
  - -- Thêm câu " Tuyên bố biến có thể bị bỏ qua. " Trong Phần 2.2
  - -- Sửa tham chiếu đến Mục 6.8 trong Phần 2.2
  - -- Thêm câu " Những từ khóa này cũng **phân biệt chữ hoa chữ thường** . " Trong mô tả **Từ khóa** trong Phần 3.3.
  - -- Sửa ví dụ trong Phần 9