

第五章 符号表管理技术

- 概述
- 符号表的组织与内容
- 非分程序结构语言的符号表组织
- 分程序结构语言的符号表组织

5.1 概述

(1) 什么是符号表?

在编译过程中,编译程序用来记录源程序中各种名字的特性信息,所以也称为名字特性表。

名 字: 程序名、过程名、函数名、用户定义类型名、变量名、常量名、枚举值名、标号名等。

特性信息: 上述名字的种类、类型、维数、参数个数、数值及目标地址 (存储单元地址) 等。

(2)建表和查表的必要性(符号表在编译过程中的作用)

- 源程序中变量要先声明，再引用。
 - 用户通过声明语句，声明各种名字，并给出它们的类型、维数等信息
 - 编译程序在处理这些声明语句时，将声明中的名字及其信息登录到符号表中，同时给变量分配存储单元，并将储单元地址登录在符号表中。
 - 当编译程序编译到引用所声明的变量时(赋值或引用其值)，要进行语法语义正确性检查(类型是否符合要求)和生成相应的目标程序，这就需要查符号表以取得相关信息。

符号表

数据区

例：int x, a, b;

...

...

L: x := a + b;

...

建表，
分配存储

x	简单变量	整型
a	简单变量	整型
b	简单变量	整型
L	标号	

1. 语法分析和语义分析
 - 说明语句、赋值语句的语法规则
 - 上下文有关分析：是否声明
 - 类型一致性检查
2. 生成目标代码
 - LOAD a的地址
 - ADD b的地址
 - STO x的地址

(3)有关符号表的操作：填表和查表

填表：当分析到程序中的说明或定义语句时，将说明或定义的名字，以及与之有关的信息填入符号表中。

例：Procedure P()

查表：

- (1) 填表前查表，检查在程序的同一作用域内名字是否重复定义；
- (2) 检查名字的种类是否与说明一致；
- (3) 对于强类型语言，要检查表达式中各变量的类型是否一致；
- (4) 生成目标指令时，要取得所需要的地址。

.....

5.2 符号表的组织与内容

(1)符号表的结构与内容

符号表的基本结构:

名字	特性(信息)

“名字”域: 存放名字，一般为标识符的符号串,也可
为指向标识符字符串的指针。

名字	特性(信息)

“特性”域：可包括多个子域，分别表示标识符的有关信息，如：

名字(标识符)的种类：简单变量、函数、过程、数组、标号、参数等

类型：如整型、浮点型、字符型、指针等

性质：变量形参、值形参等

值：常量名所代表的数值

地址：变量所分配单元的首址或地址位移

大小：所占的字节数

作用域的嵌套层次：

对于数组：维数、上下界值、计算下标变量地址所用的信息（数组信息向量）以及数组元素类型等。

对于记录（结构、联合）：域的个数，每个域的域名、地址位移、类型等。

对于过程或函数：形参个数、所在层次、函数返回值类型、局部变量所占空间大小等。

对于指针：所指对象类型等。

(2)符号表的组织方式

1.统一符号表:不论什么名字都填入统一格式的符号表中

符号表表项应按信息量最大的名字设计,填表、查表比较方便,结构简单,但是浪费大量空间。

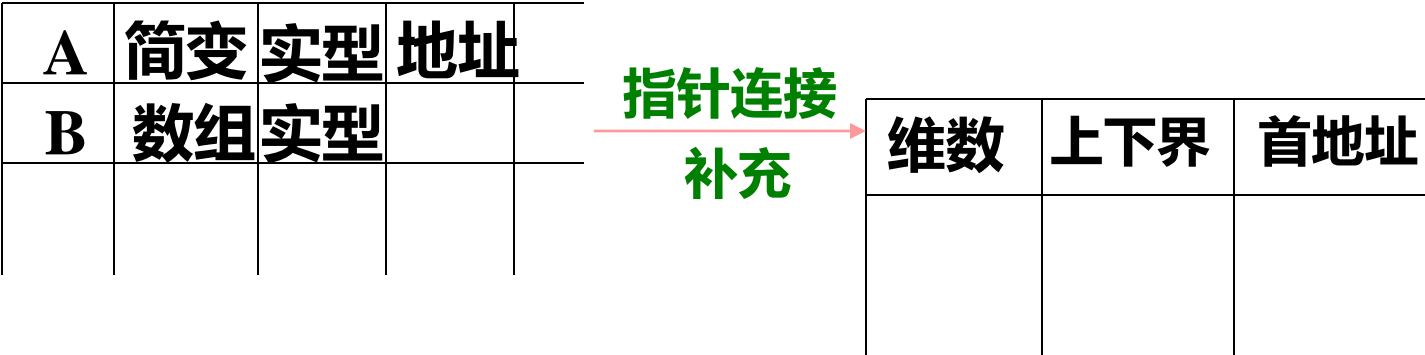
2.对于不同种类的名字分别建立各种符号表

节省空间,但是填表和查表不方便。

3.折中办法:大部分共同信息组成统一格式的符号表,特殊信息另设附表,两者用指针连接。

```

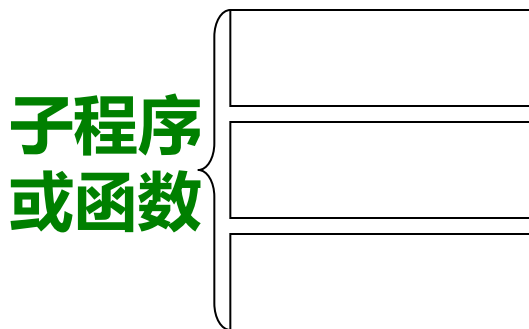
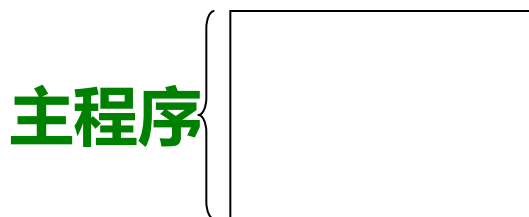
例: begin
      A : real;
      B : array [1:100] of real;
      :
      :
    end
  
```



5.3 非分程序结构语言的符号表组织

(1)非分程序结构语言: 每个可独立进行编译的程序单元是一个不包含有子模块的单一模块, 如FORTRAN语言。

FORTRAN程序构造



主程序和子程序中可
定义common语句

(2)标识符的作用域及基本处理办法

1. 作用域: **全局**:子程序名,函数名和公共区名。
局部: 程序单元中定义的变量。

2. 符号表的组织:

全局符号表
局部符号表

3. 基本处理办法:

<1> 子程序、函数名和公共区名填入全局符号表。

<2> 在子程序（函数）声明部分读到标识符，
造局部符号表。

查本程序单元局部符号表，有无同名？

有,重复声明,报错

无,造表

<3> 在语句部分读到标识符,查表:

查本程序单元局部符号表，有无同名？

有,即已声明

无,查全局变量表

有,全局量

无,无定义标识符

4. 程序单元结束: 释放该程序单元的局部符号表。
5. 程序编译完成: 释放全部符号表。

(3)符号表的组织方式

1. 无序符号表: 按扫描顺序建表,查表要逐项查找

查表操作的平均长度为 $n+1/2$

2. 有序符号表：符号表按变量名进行字典式排序

线性查表： $n+1/2$

折半查表： $\log_2 n - 1$

3. 散列符号表(Hash表)：符号表地址 = Hash(标识符)

解决：冲突

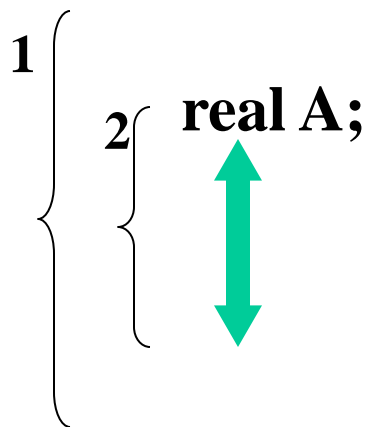
5.4 分程序结构语言的符号表组织

(1) 分程序结构语言:模块内可嵌入子模块

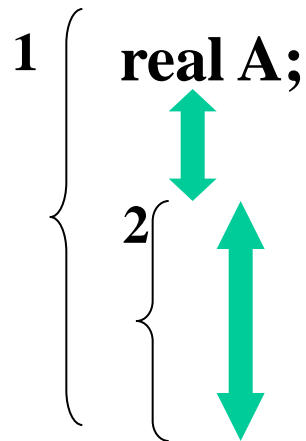
(2) 标识符的作用域和基本处理方法 :

作用域：标识符局部于所定义的模块(最小模块)

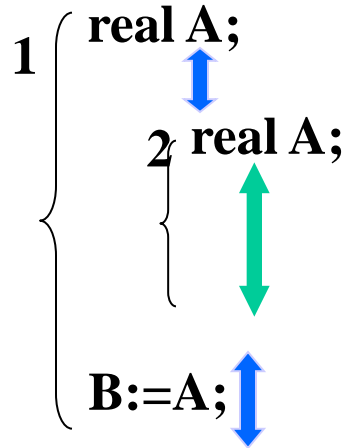
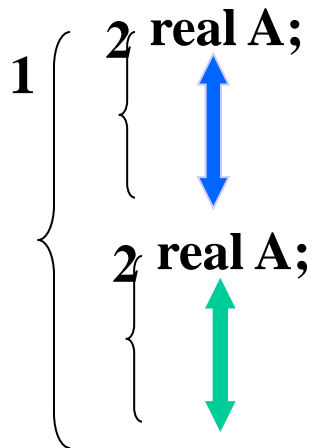
- ① 模块中所定义的标识符作用域是定义该标识符的子程序



A为内分程序局部变量



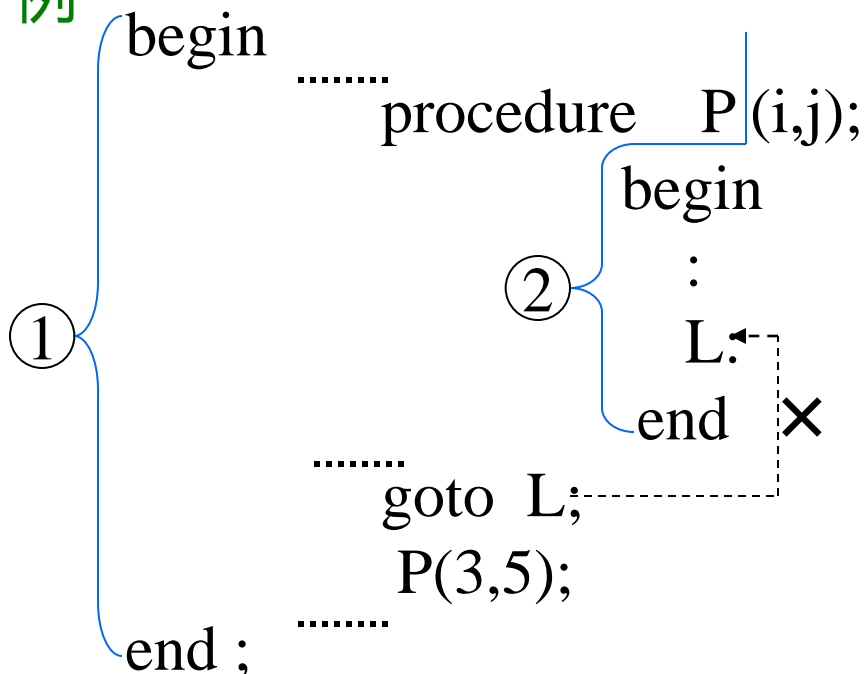
A为可作用于内分程序的全局变量



都是局部变量

- ② 过程或函数说明中定义的标识符(包括形参)其作用域为本过程体。

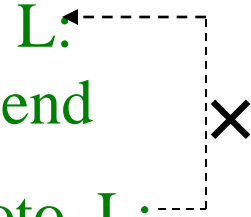
例



- ③ 循环语句中定义的标识符,其作用域为该循环语句。

```

for ... .. do
  begin
    :
    L:
  end
  Goto L;
  :
  
```



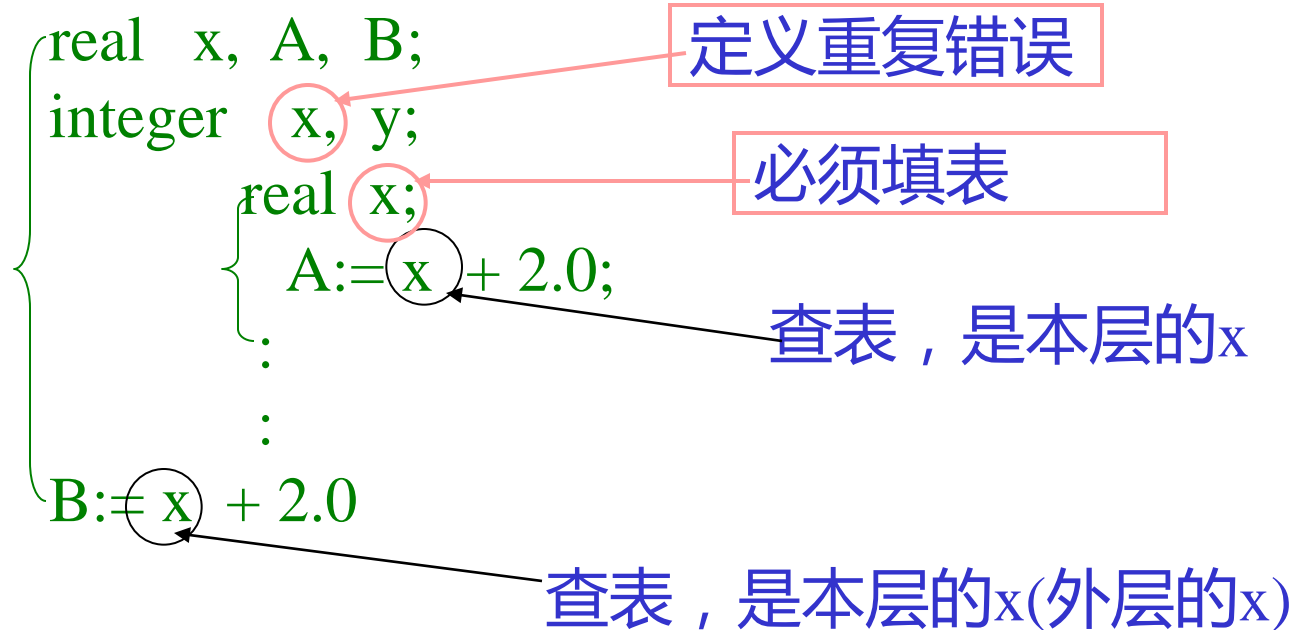
不能从循环体外转到循环体内。循环语句应看作一层

基本处理办法:

建查符号表均要遵循标识符的作用域规定进行。

建表：不能重复,不能遗漏

查表：按标识符作用域



处理方法:

a. 在程序声明部分读到标识符时(声明性出现),建表:

查本层符号表,有无同名?
 有,重复声明,报错
 无,填入符号表

b. 在语句中读到标识符(引用性出现),查表:

查本层符号表,有无同名?
 有,即已声明,取该名字信息 (局部量)
 是,未声明标识符,报错
 无,是否是最外层?
 否,转到直接外层
 (n-1)

c. 标准标识符的处理

主要是语言定义的一些标准过程和函数的名字，它们是标识符的子集。

如 `sin` `con` `abs`....

特点：1) 用户不必声明,就可全程使用

2) 设计编译程序时，标准名字及其数目已知

处理方法：1) 单独建表：使用不便，费时。

2) 预先将标准名填入名字表中

最外层

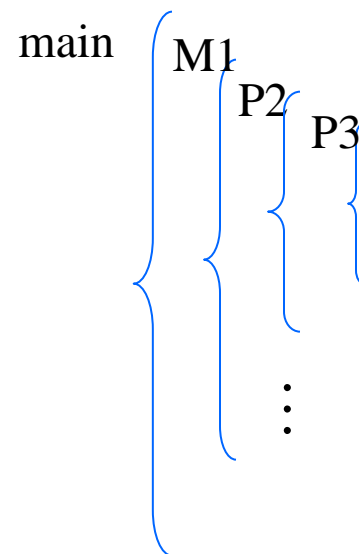
例:Pascal程序的分程序结构示例如下:

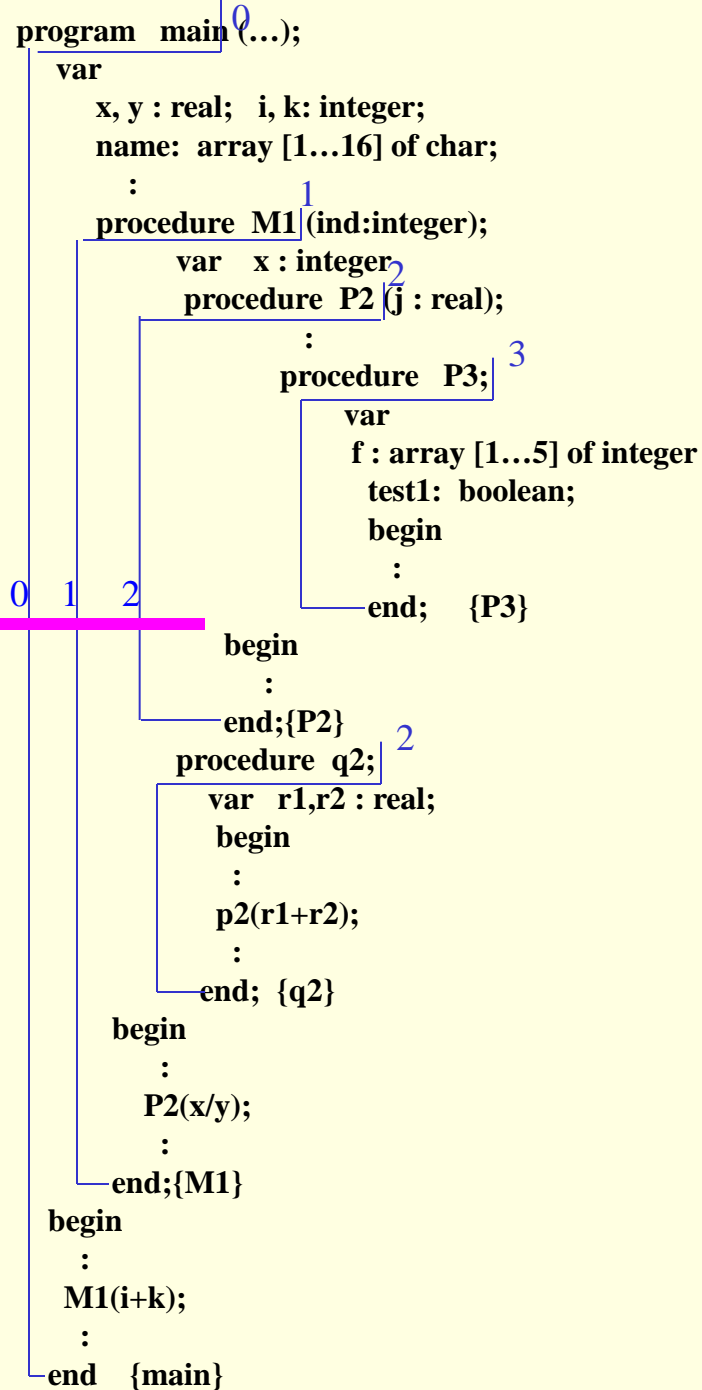
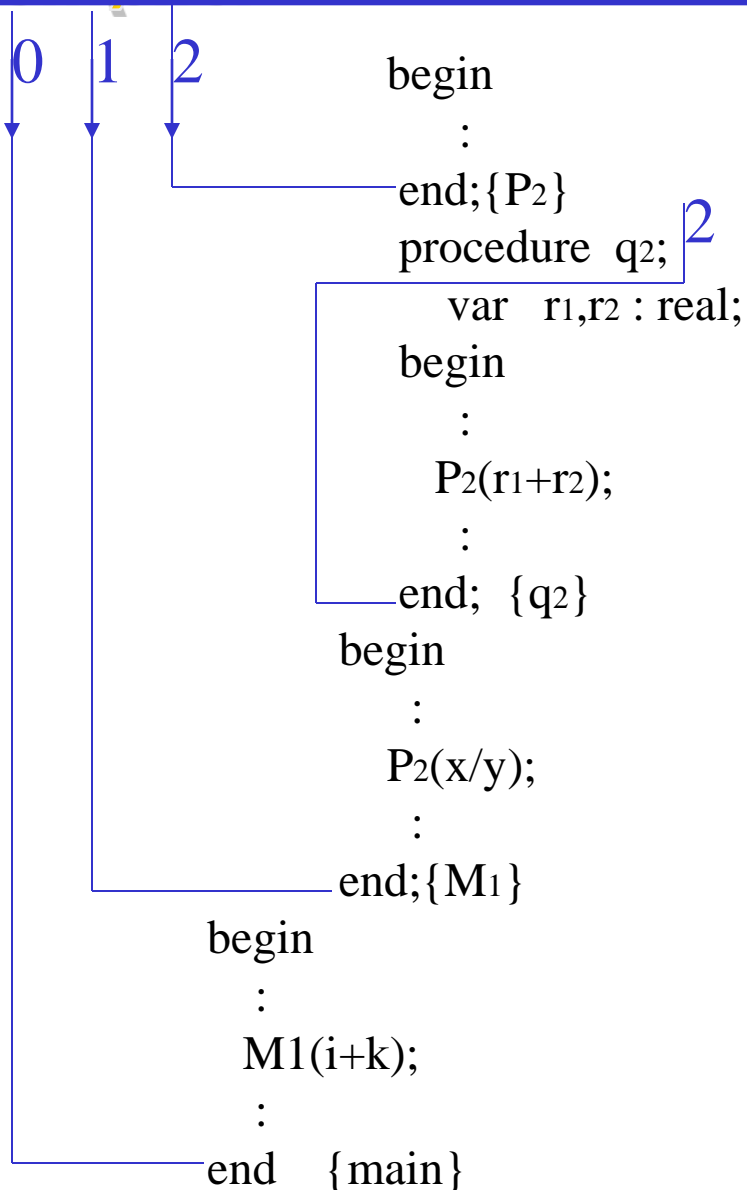
```

program main0(...);
  var
    x, y : real; i, k: integer;
    name: array [1...16] of char;
    :
1  procedure M1(ind:integer);
2    var x : integer;
3    procedure P2(j : real);
      :
      procedure P3;
        var
          f : array [1...5] of integer;
          test1: boolean;
          begin
            :
            end; {P3}
      end;
    end;
  end;
end;

```

<程序> ::= program <标识符> (<标识符>{, <标识符>}) <分程序>.
 <分程序> ::= [<常量说明部分>][<变量说明部分>][<过程说明部分>]<语句>
 <常量说明部分> ::= const<常量定义>{,<常量定义>;}
 <常量定义> ::=
 <无符号整数> ::= <数字>{<数字>}
 <标识符> ::= <字母>{<字母>|<数字>}
 <变量说明部分> ::= var<标识符>{,<标识符>}:<类型>;
 <过程说明部分> ::= <过程首部><分程序>{;<过程说明部分>;}
 <过程首部> ::= procedure<标识符>(<参数表>;

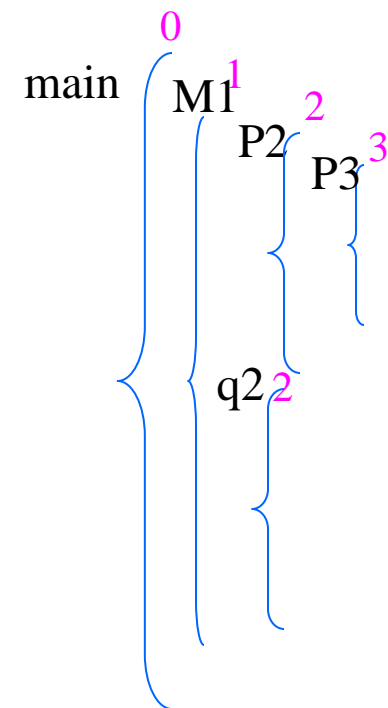


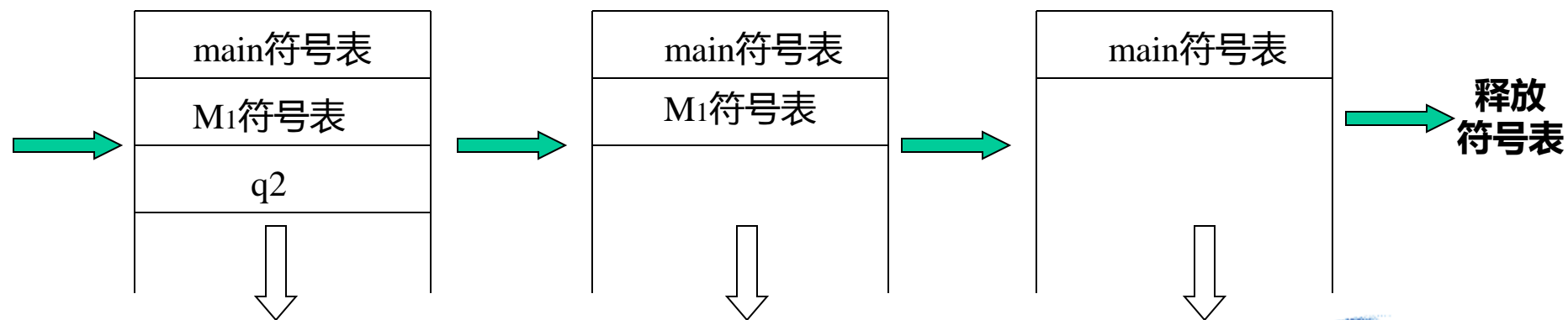
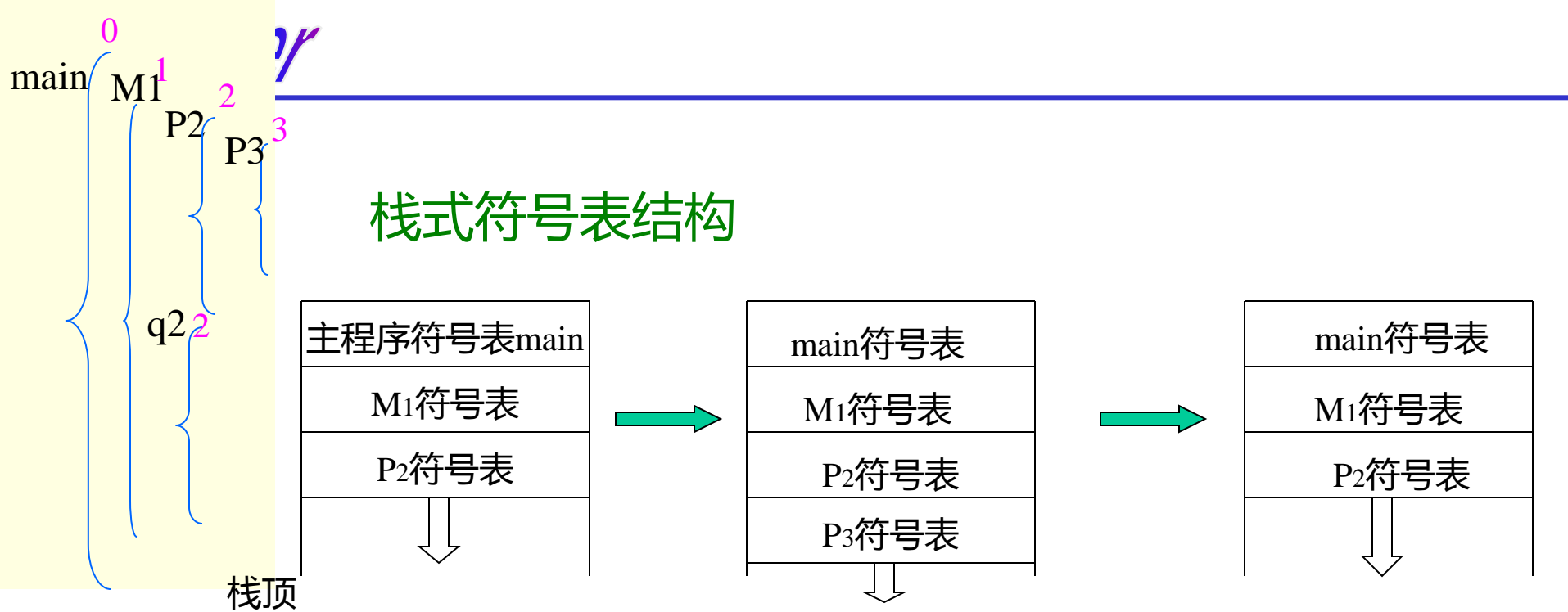



```

program main0(...);
var
  x, y : real;  i, k: integer;
  name: array [1...16] of char;
  :
  procedure M11(ind:integer);
  var  x : integer2;
  procedure P22(j : real);
  :
  procedure P33;
  var
    f : array [1...5] of integer
    test1: boolean;
    begin
      :
    end; {P3}
  begin
    :
  end;{P2}
  procedure q22;
  var  r1,r2 : real;
  begin
    :
    P2(r1+r2);
    :
  end; {q2}
  begin
    :
    P2(x/y);
    :
  end;{M1}
begin
  :
  M1(i+k);
  :
end {main}

```





	name	kind	type	lev	other inf
1	x	var	real	0	
2	y	var	real	0	
3	i	var	int	0	
4	k	var	int	0	
5	name	var	array	0	
6	M ₁	proc		0	
7	ind	para	int	1	
8	x	var	int	1	
9	P ₂	proc		1	
10	j	para	real	2	
11	P ₃	proc		2	
12	f	var	array	3	
13	test1	var	boolean	3	

```

program main0(...);
  var
    x, y : real; i, k: integer;
    name: array [1...16] of char;
    :
    procedure M11(ind:integer);
      var x : integer2;
      procedure P22(j : real);
        :
        procedure P33;
          var
            f : array [1...5] of integer;
            test1: boolean;
            begin
              :
            end; {P3}
      end;
    end;
  end;

```

分程序索引表

0	1
1	7
2	10
3	12

main

M₁

P₂

P₃

	name	kind	type	lev	other inf
1	x	var	real	0	
2	y	var	real	0	
3	i	var	int	0	
4	k	var	int	0	
5	name	var	array	0	
6	M ₁	proc		0	
7	ind	para	int	1	
8	x	var	int	1	
9	P ₂	proc		1	
10	j	para	real	2	
11	P ₃	proc		2	
12	f	var	array	3	
13	test1	var	boolean	3	

```

program main0(...);
var
  x, y : real; i, k: integer;
  name: array [1...16] of char;
  :
1  procedure M1(ind:integer);
2    var x : integer;
3    procedure P2(j : real);
      :
      procedure P3;
          var
            f : array [1...5] of integer;
            test1: boolean;
            begin
              :
            end; {P3}
      begin
        :
      end;{P2}
2    procedure q2;
      var r1,r2 : real;
      begin
        :
        P2(r1+r2);
        :
      end; {q2}
    begin
      :
      P2(x/y);
      :
    end;{M1}
  begin
    :
    M1(i+k);
    :
  end {main}
  
```

	name	kind	type	lev	other inf
1	x	var	real	0	
2	y	var	real	0	
3	i	var	int	0	
4	k	var	int	0	
5	name	var	array	0	
6	M ₁	proc		0	
7	ind	para	int	1	
8	x	var	int	1	
9	P ₂	proc		1	
10	j	para	real	2	
11	P ₃	proc		2	

```

program main0(...);
var
  x, y : real; i, k: integer;
  name: array [1...16] of char;
  :
1  procedure M1(ind:integer);
    var x : integer;
    2  procedure P2(j : real);
      :
      3  procedure P3;
        var
          f : array [1...5] of integer;
          test1: boolean;
          begin
            :
            end; {P3}
      begin
        :
        end;{P2}
    2  procedure q2;
      var r1,r2 : real;
      begin
        :
        P2(r1+r2);
        :
        end; {q2}
      begin
        :
        P2(x/y);
        :
        end;{M1}
    begin
      :
      M1(i+k);
      :
    end {main}
  
```

编译 q_2 说明部分后:

7	ind	para	int	1	M ₁
8	x	var	int	1	
9	P ₂	proc		1	
10		para	real	1	
11	q ₂	proc		1	
12	r ₁	var	real	2	q ₂
13	r ₂	var	real	2	

```

program main0(...);
var
  x, y : real; i, k: integer;
  name: array [1...16] of char;
  :
  procedure M11(ind:integer);
    var x : integer;
    procedure P22(j : real);
      :
      procedure P33;
        var
          f : array [1...5] of integer;
          test1: boolean;
          begin
            :
          end; {P3}
      begin
        :
      end;{P2}
    procedure q22;
      var r1,r2 : real;
      begin
        :
        P2(r1+r2);
        :
      end; {q2}
    begin
      :
      P2(x/y);
      :
    end;{M1}
  begin
    :
    M1(i+k);
    :
  end {main}
  
```

编译完 q_2 过程体:

7	ind	para	int	
8	x	var	int	
9	P_2	proc		
10		para	real	
11	q_2	proc		

当过程和函数体编译完成后，
应将与之相应的 参数名和局部变量名
以及后者的特性信息从符号表中删去。

要求：给出一段程序，会画出其栈式符号表

作业：P115-116 3,5

```

program main( $q_2$ );
var
  x, y : real; i, k: integer;
  name: array [1...16] of char;
  :
  procedure M1( $ind$ :integer);1
    var x : integer;2
    procedure P2( $j$  : real);2
      :
      procedure P3;3
        var
          f : array [1...5] of integer;
          test1: boolean;
          begin
            :
            end; {P3}
          begin
            :
            end;{P2}
          procedure  $q_2$ ;2
            var r1,r2 : real;
            begin
              :
              P2(r1+r2);
              :
              end; { $q_2$ }
            begin
              :
              P2(x/y);
              :
              end;{M1}
            begin
              :
              M1(i+k);
              :
              end {main}
  
```