# 第八章 符号表

同济大学计算机系

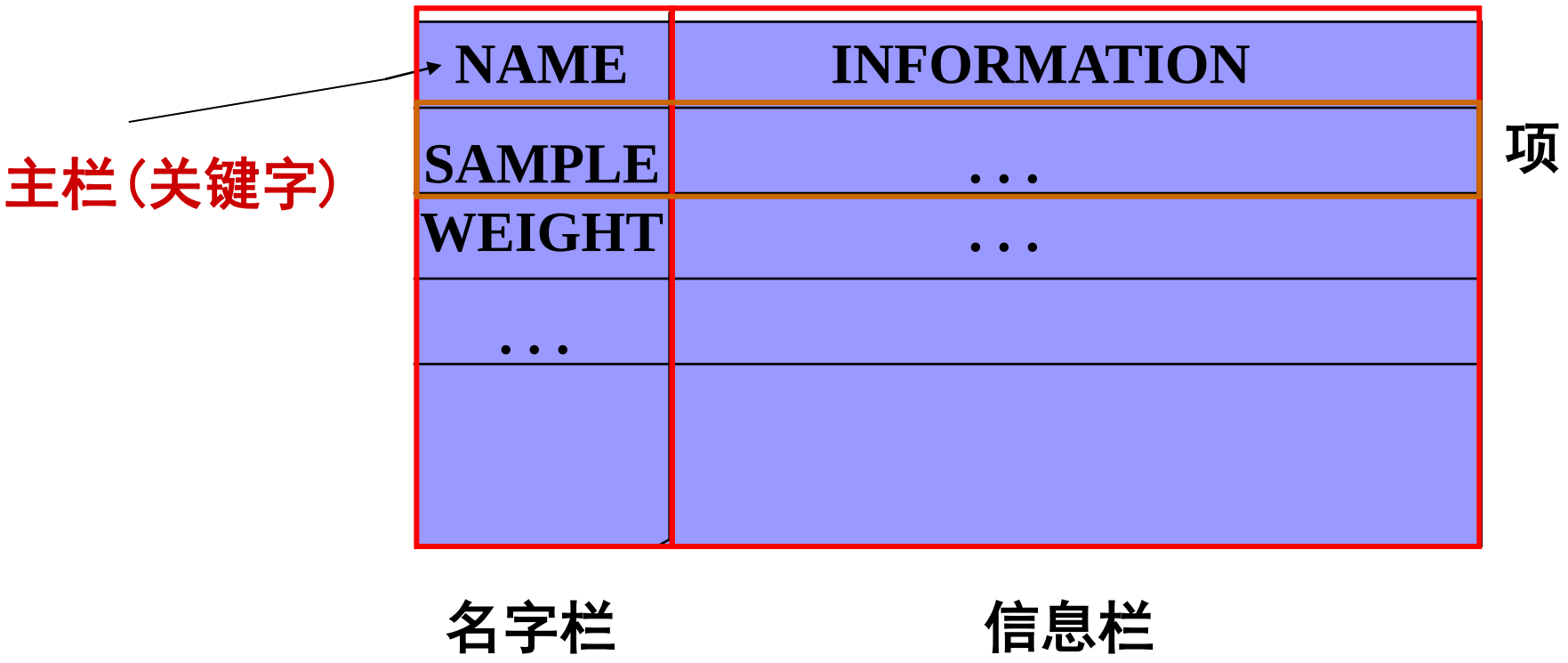
# 内容线索

- **符号表的组织与作用**
- **名字的作用范围**
- **符号表的内容**

# 符号表的作用

- 收集信息
  - 源程序中的各种名字及其属性、特征等。
- 提供使用
  - 上下文语义的合法检查的依据；
  - 目标代码生成阶段地址分配的依据等。

# 符号表的组织与使用



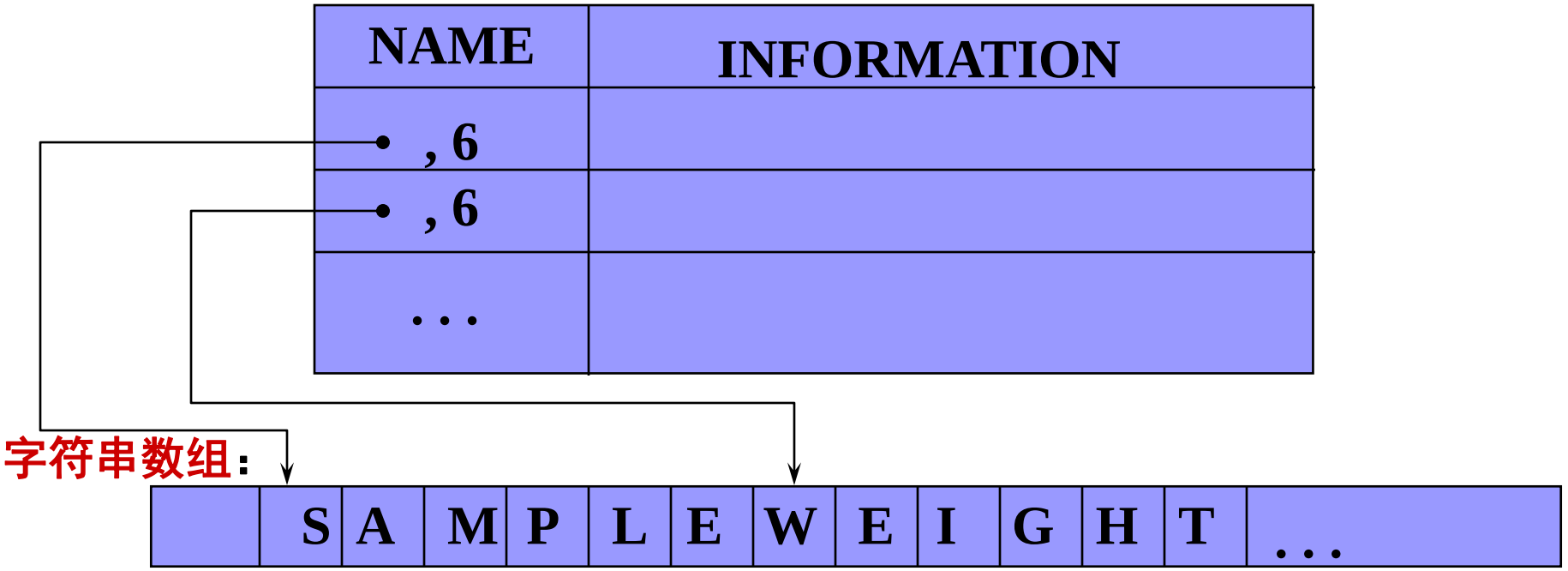
# 符号表的操作

- 填入: 填入新名字、信息.
- 查找: 给出名字，确定它是否在表中.
- 访问: 给出名字，访问有关的信息.
- 更新: 给出名字，更新有关信息.
- 删除: 删除一个或一组记录.

# 符号表的组织方式

- 直接方式：各栏长度固定，内容直接填入
  - 优点：易于组织、填写和查找
  - 缺点：浪费空间
- 间接方式：内容填入其他数据结构中，符号表栏中仅放置指示器，指向该位置.

# 名字栏的间接组织方式

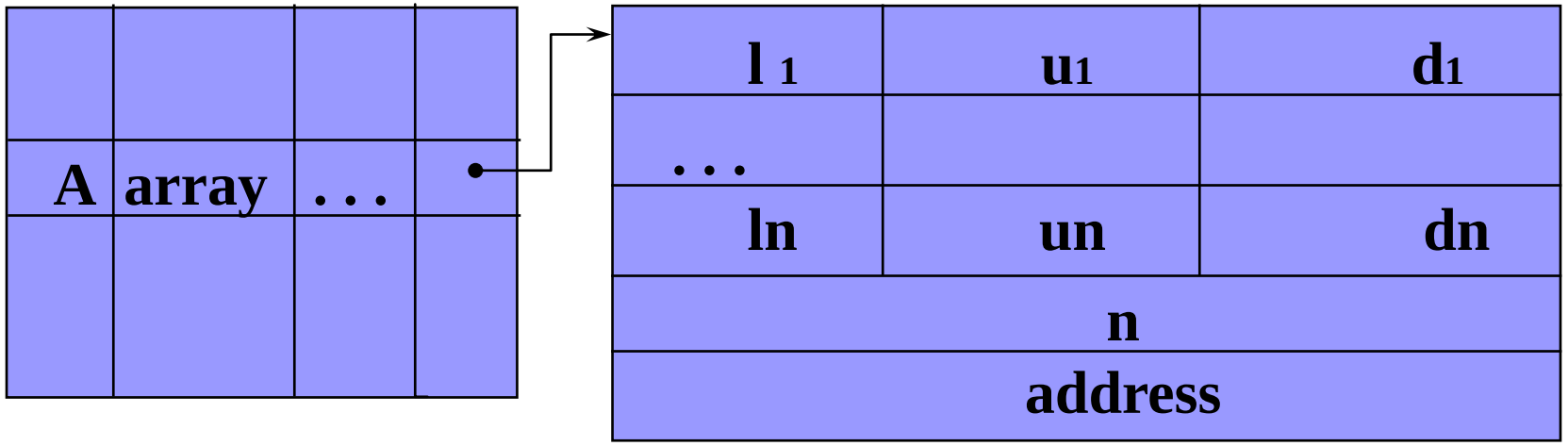



- 指针（指示器）：标识符的起始位置
- 整数：标识符的长度

# 信息栏的间接组织方式

- 信息栏也可用间接组织方式，如数组标识符的**内情向量表**

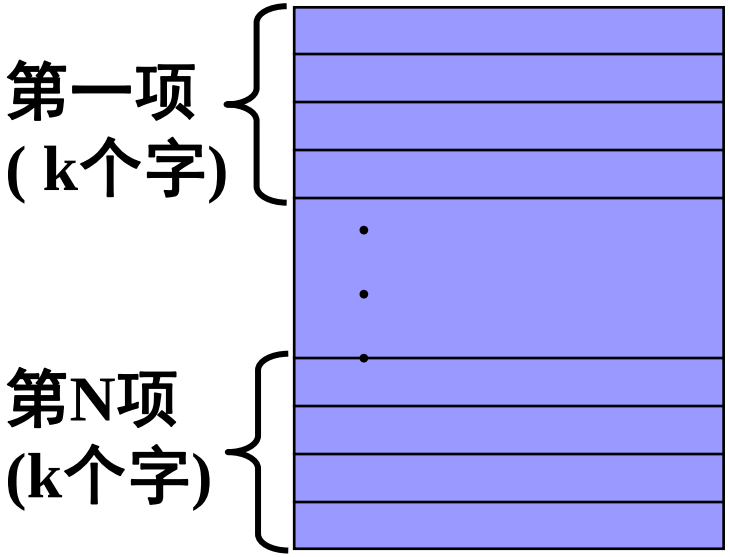
**array A [ l1 :u1 , l2: u2 , . . . , ln:un ]:type;**



# 符号表的存储

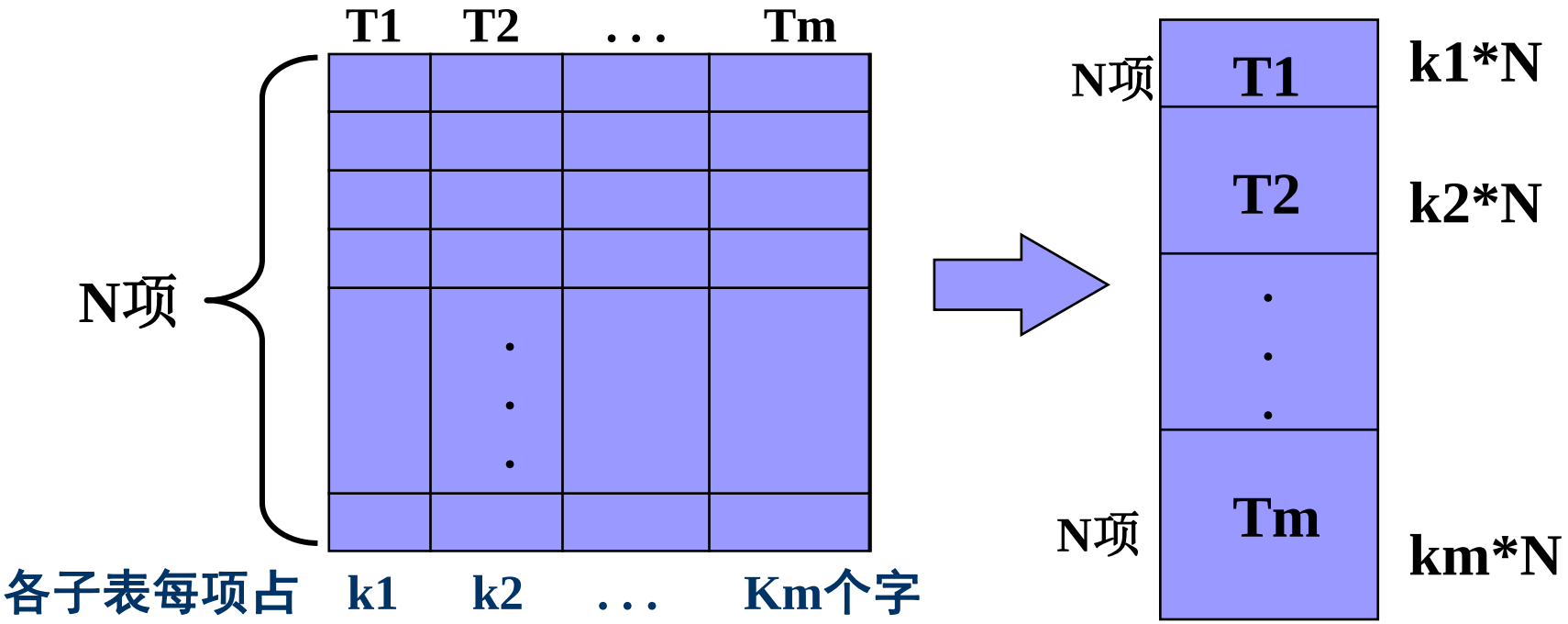
设含N项的符号表，每项K个字

- 连续存储
  - 把每一项置于连续K存储单元中，构成一张K*N的表



# 符号表的存储

- 子表存放
  - 把整个符号表分成m个子表，如$T_1,T_2,\ldots T_m$，每个子表含有N项.
  - 第i项的全部内容是$T_1[i]\ldots T_m[i]$的并



# 符号表的种属

- 按名字的不同种属建立多张符号表，如常数表、变量名表、过程名表、…

例．PASCAL程序段：

```
PROCEDURE INCWAP(M，N:INTEGER);
LABEL  START;
VAR
  K:INTEGER;
BEGIN
START:
      K:=M+1;
      M:=N+4;
      N:=K;
END.
```

```
PROCEDURE INCWAP(M，N:INTEGER);
LABEL  START;
VAR
  K:INTEGER;
BEGIN
START:
    K:=M+1;
    M:=N+4;
    N:=K;
END.
```

表 0.1 符号名表 SNT

| NAME | INFORMATION |
|---|---|
| M | 形式参数，整型，值参数 |
| N | 形式参数，整型，值参数 |
| K | 整型，变量 |

```
PROCEDURE INCWAP(M，N:INTEGER);
LABEL  START;
VAR
  K:INTEGER;
BEGIN
START:
    K:=M+1;
    M:=N+4;
    N:=K;
END.
```

表 0.2 常数表 CT

| | 值 (VALUE) |
|---|---|
| (1) | 1 |
| (2) | 4 |

```
PROCEDURE INCWAP(M，N:INTEGER);
LABEL  START;
VAR
  K:INTEGER;
BEGIN
START:
    K:=M+1;
    M:=N+4;
    N:=K;
END.
```

表 0.3 入口名表 ENT

| | NAME | INFORMATION |
|---|---|---|
| (1) | INCWAP | 二目子程序,<br>入口四元式:1 |

```
PROCEDURE INCWAP(M，N:INTEGER);
LABEL  START;
VAR
  K:INTEGER;
BEGIN
START:
    K:=M+1;
    M:=N+4;
    N:=K;
END.
```

表 0.4　标号表 LT

| | NAME | INFORMATION |
|---|---|---|
| (1) | START | 四元式:(4) |

表 0.1 符号名表 SNT

| NAME | INFORMATION |
|---|---|
| M | 形式参数，整型，值参数 |
| N | 形式参数，整型，值参数 |
| K | 整型，变量 |

表 0.2 常数表 CT

| | 值(VALUE) |
|---|---|
| (1) | 1 |
| (2) | 4 |

表 0.3 入口名表 ENT

| | NAME | INFORMATION |
|---|---|---|
| (1) | INCWAP | 二目子程序,入口四元式:1 |

表 0.4 标号表 LT

| | NAME | INFORMATION |
|---|---|---|
| (1) | START | 四元式:(4) |

```
PROCEDURE INCWAP(M，N:INTEGER);
LABEL  START;
VAR
  K:INTEGER;
BEGIN
START:
    K:=M+1;
    M:=N+4;
    N:=K;
END.
```

表 0.5　四元式表 QT

| | OPR | OPN1 | OPN2 | RESULT |
|---|---|---|---|---|
| (1) | link | | | |
| (2) | par | INCWAP | 1 | M |
| (3) | par | INCWAP | 2 | N |
| (4) | + | M | 1 | K |
| (5) | + | N | 4 | M |
| (6) | := | K | | N |
| (7) | return | | | |

# 内容线索

- ✓ 符号表的组织与作用
- ■ 名字的作用范围
- ■ 符号表的内容

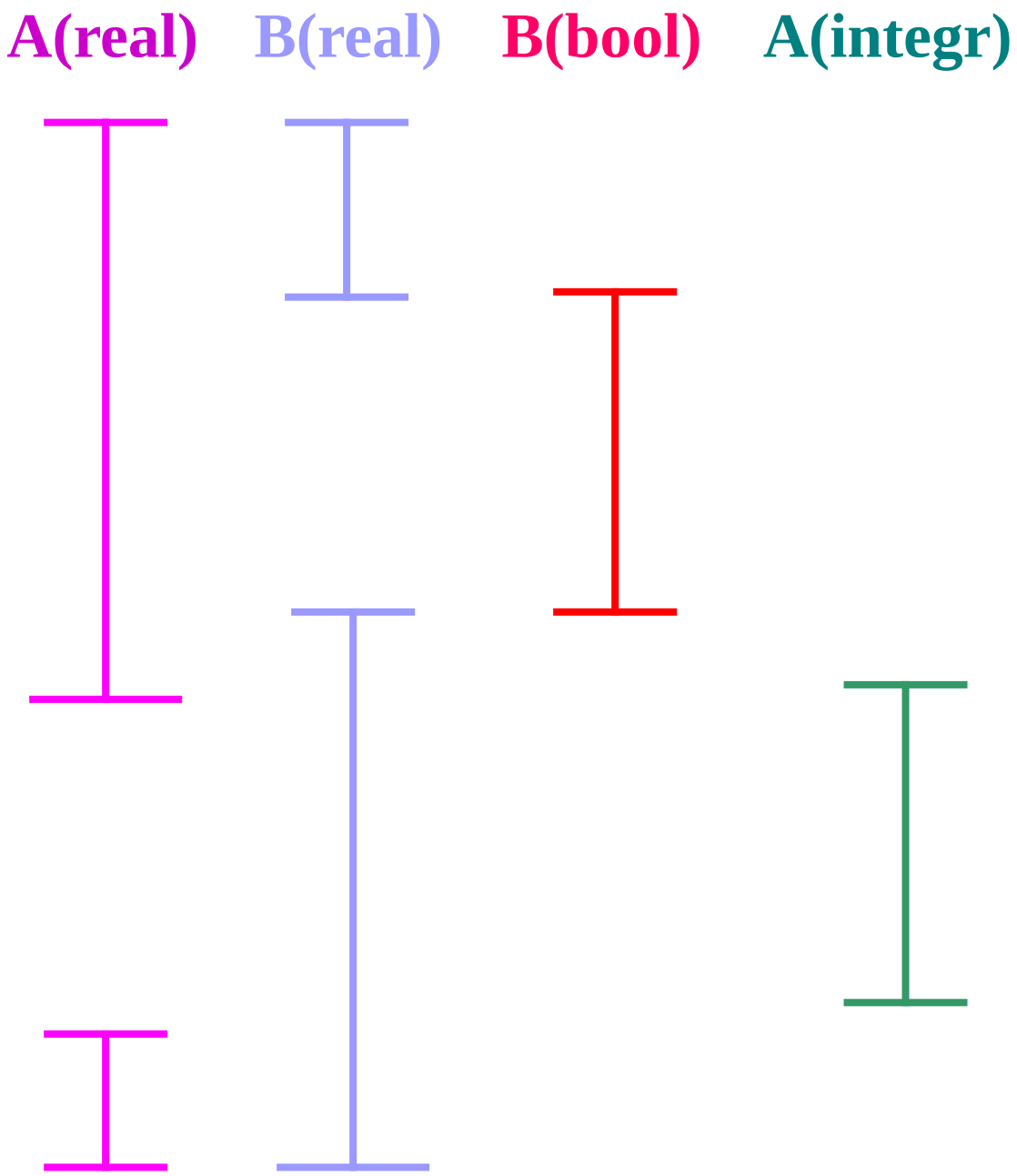
# 名字的作用范围

- 在许多程序语言中,名字都有一个确定的作用范围.
- 两种程序体结构
  - 单层（并列）结构，如FORTRAN
    - 一个FORTRAN程序由一个主程序段和若干个辅程序段组成
  - 多层（嵌套）结构，如PASCAL，ADA
    - 过程可以嵌套和递归

# 作用域

- **作用域**：一个名字能被使用的区域范围称作这个名字的作用域。
- 允许同一个标识符在不同的过程中代表不同的名字。
- 名字作用域规则—“最近嵌套原则”
  - 一个在子程序B1中说明的名字X只在B1中有效（局部于B1）；
  - 如果B2是B1的一个内层子程序且B2中对标识符X没有新的说明，则原来的名字X在B2中仍然有效。如果B2对X重新作了说明，那么，B2对X的任何引用都是指重新说明过的这个X。

```
program main
   var  A, B : real;
   …
   procedure P1
    var B:boolean;
          …
   begin
          …
   end
   procedure P2
    var A:integer;
          …
   begin
          …
   end
begin
   …
end
```



# 作用域的实现

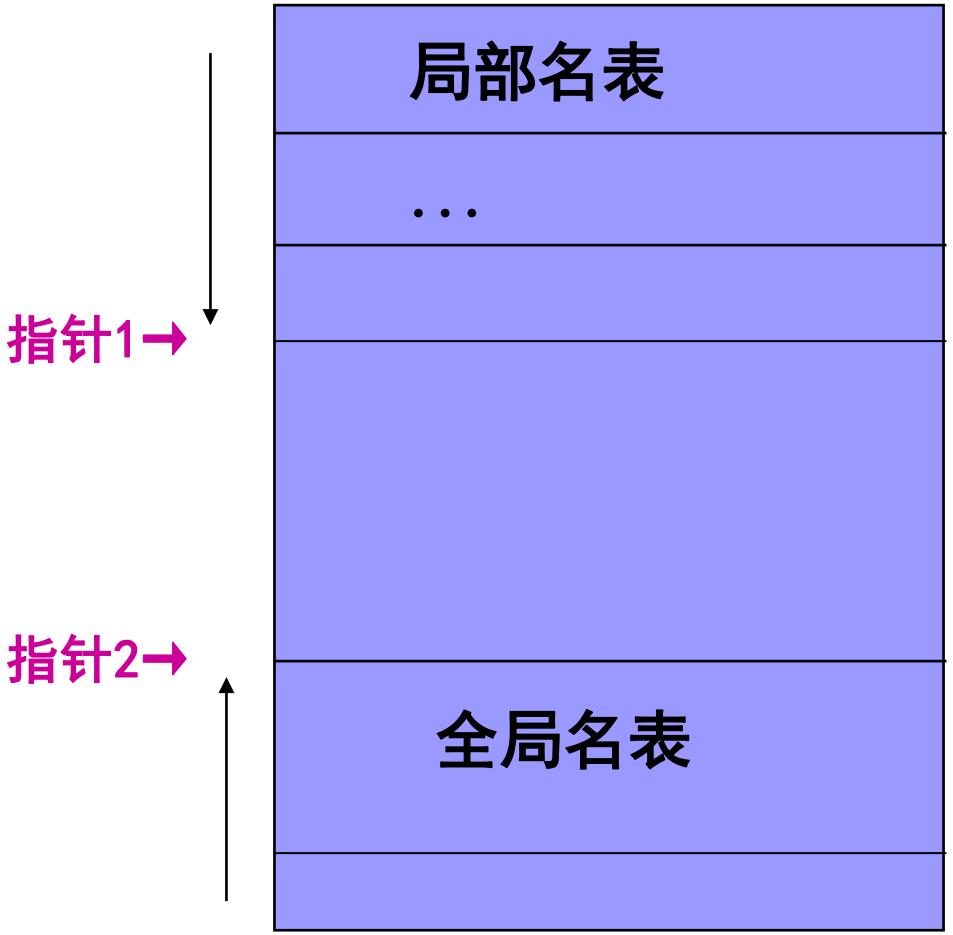
- **两种做法：**
  - **引入“过程编号”属性：**

    **〈名字，过程编号〉**

    **查找时，先查找本过程编号的名字，查不到则查找外层过程编号的名字，…，等等.**
  - **按”栈”式思想组织符号表。查找时，从后往前查找，碰到的第一个名字就是所需查找的名字.**

# FORTRAN的符号表组织

- 变量、数组和语句函数名的作用范围就是他们所处的程序段
- 把局部名和全局名分别存在不同的地方
- 一遍扫描时，当一段程序处理完后，其局部名不需再保留.

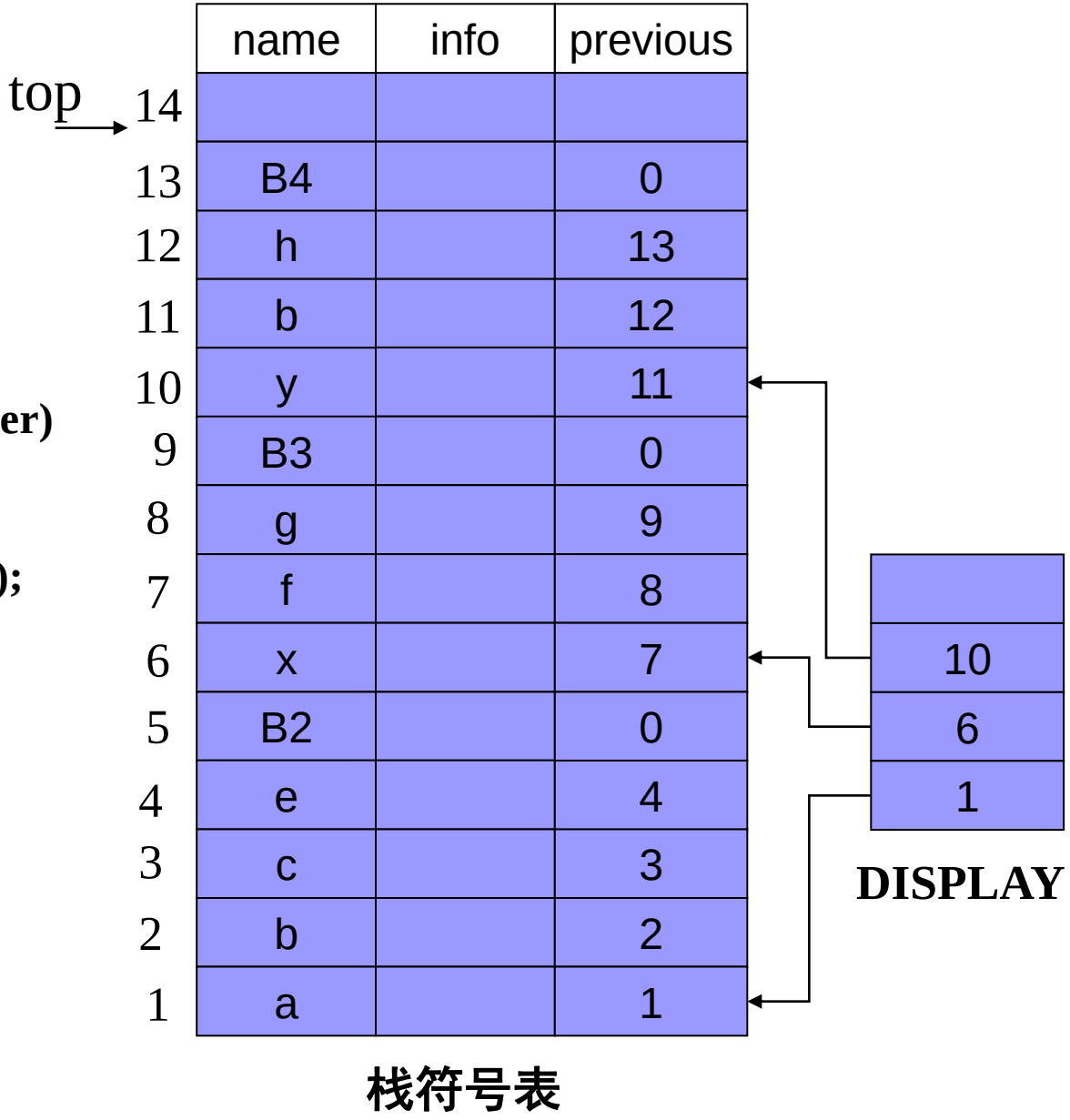


# Pascal的符号表组织

- 符号表设计为**栈符号表**，新的名字出现总是从栈顶填入
  - top：栈顶指针
  - 信息栏指针域previous：指明同一层中前一名字的位置
- **显示层次关系表**(嵌套层次表,display表)，存放各嵌套过程子表在主表起始位置

例.program B1(input,output)

```
  const a=10
  var b,c: integer;  e: real;
  procedure B2(x:real)
    var f,g:real;
    procedure B3(y:real)
      const b=5;
      var h:boolean;
      procedure B4(z:integer)
        var i: char;
        begin…
          if e<0 then B3(f);
        … end;
      begin  …
        B4(a);
      …  end;
    begin  …
      B3(c);
    …   end;
  begin  …
    B2(e);
  …   end
```



# 内容线索

- ✓ 符号表的组织与作用
- ✓ 名字的作用范围
- ■ 符号表的内容

# 符号表的内容

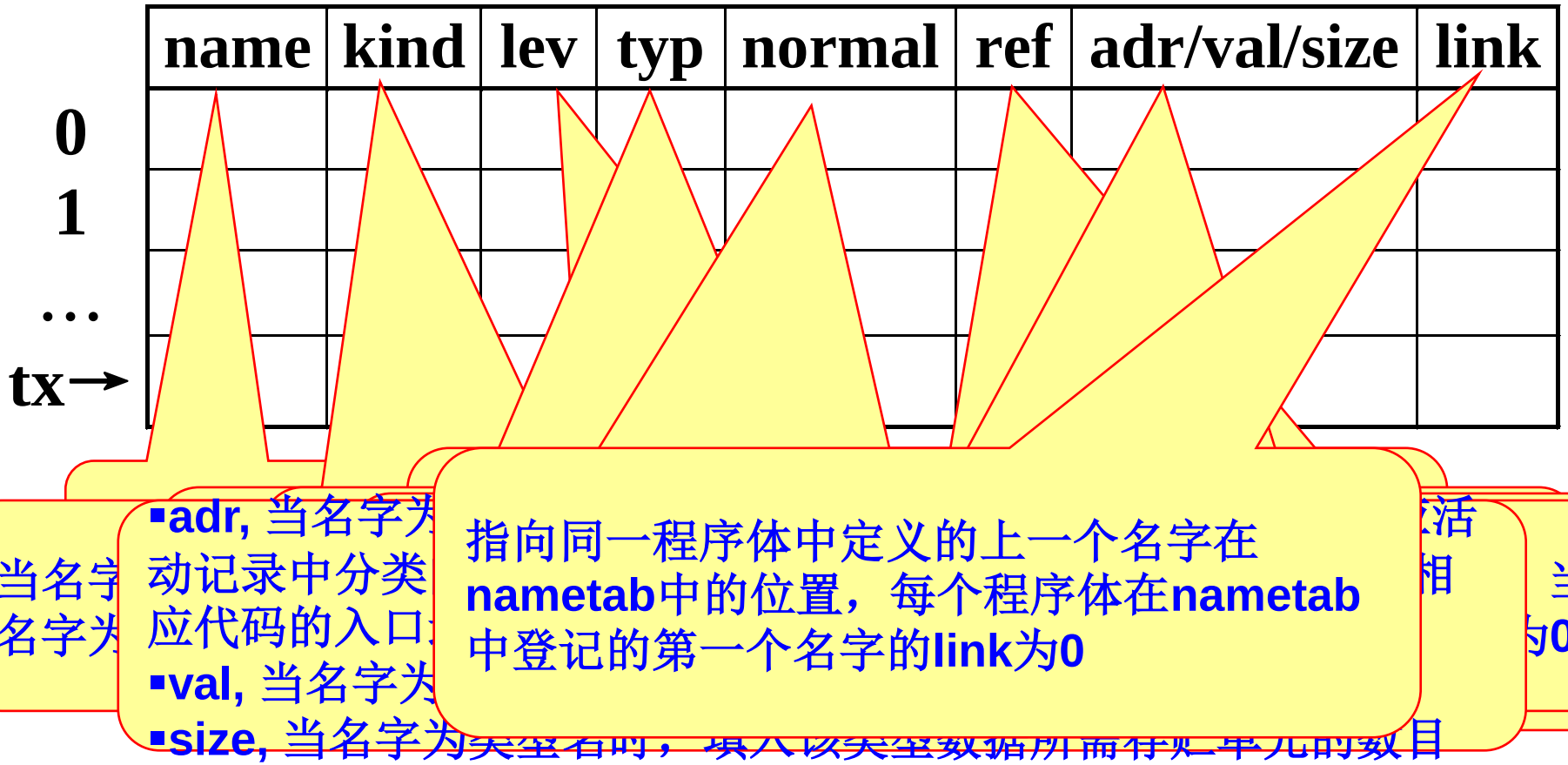
- 符号表的信息栏中登记了每个名字的有关性质
  - 类型：整、实或布尔等
  - 种属：简单变量、数组、过程等
  - 大小：长度，即所需的存储单元字数
  - 相对数：指分配给该名字的存储单元的相对地址

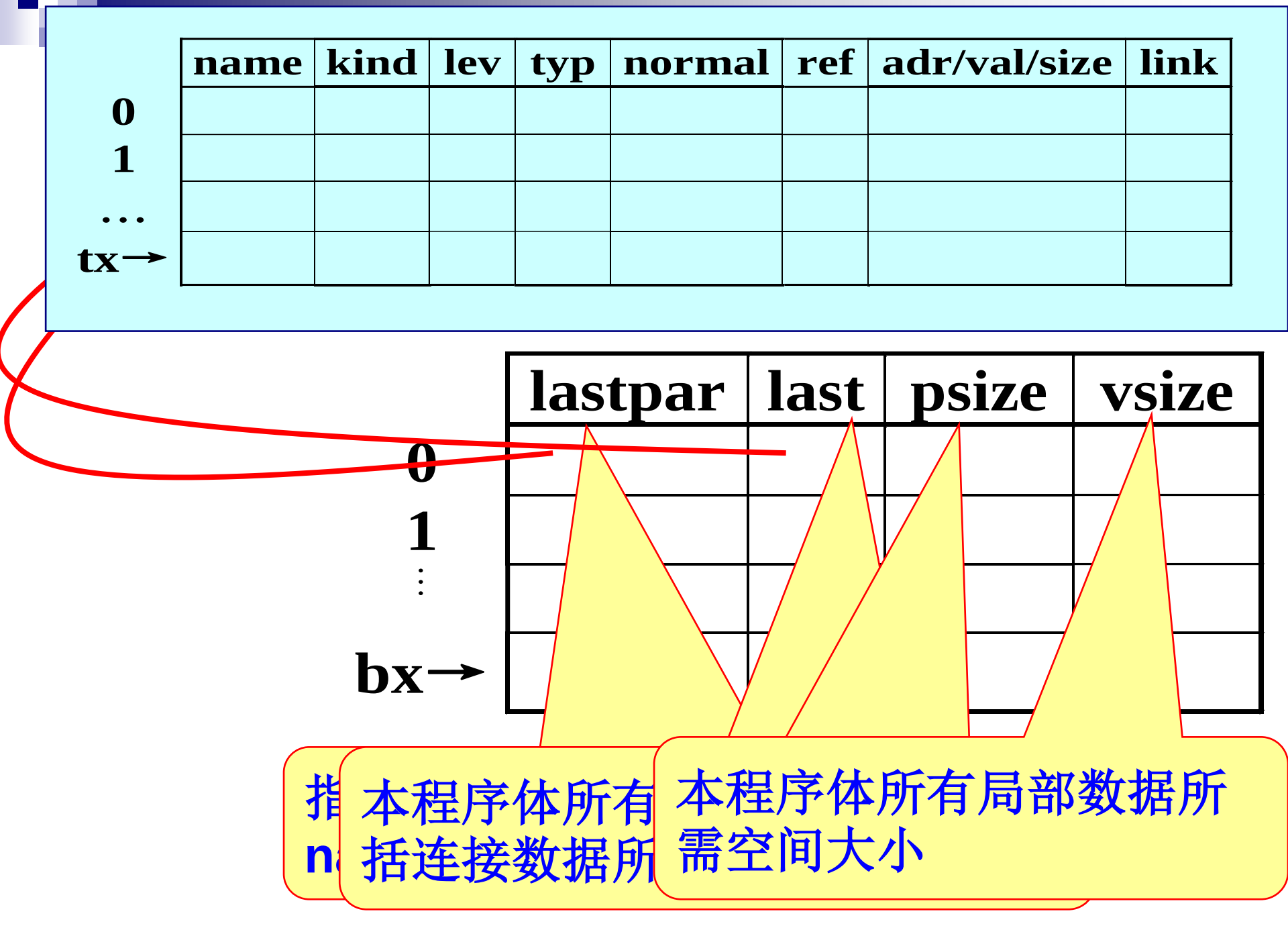
# 附：PL 语言编译程序的符号表

- 名字表(nametab)
- 程序体表(btab)
- 层次显示表(display)
- 数组信息表(atab)
- 中间代码表(code)

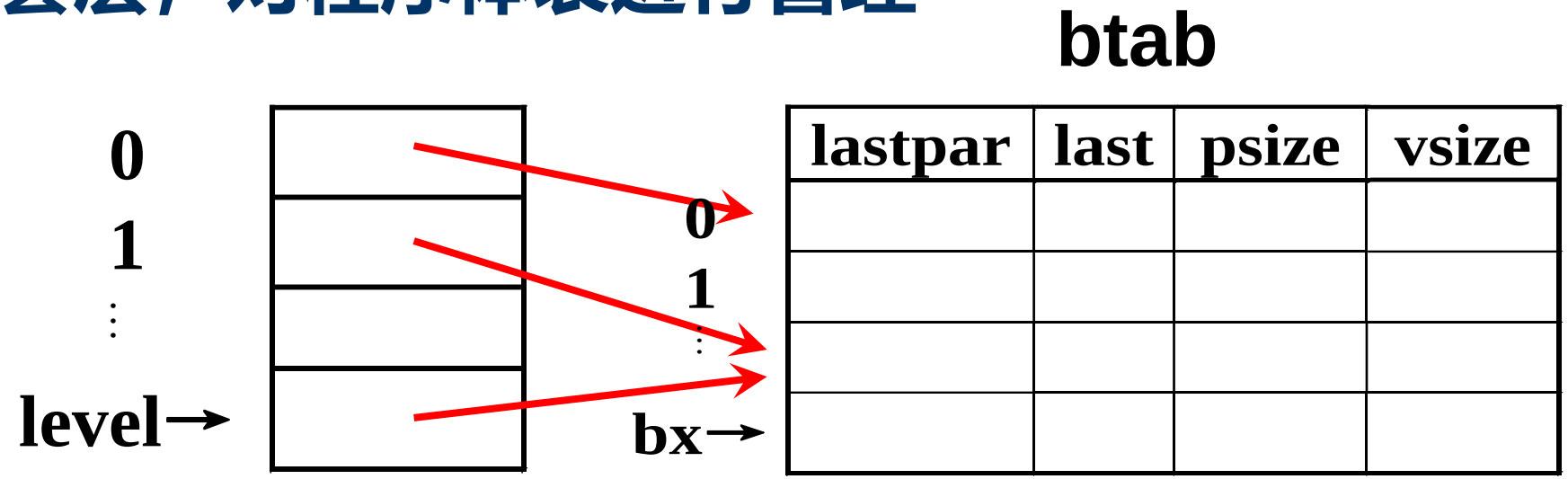
# 1) 名字表(nametab)

## 名字表nametab：登记程序中出现的各种名字及其属性

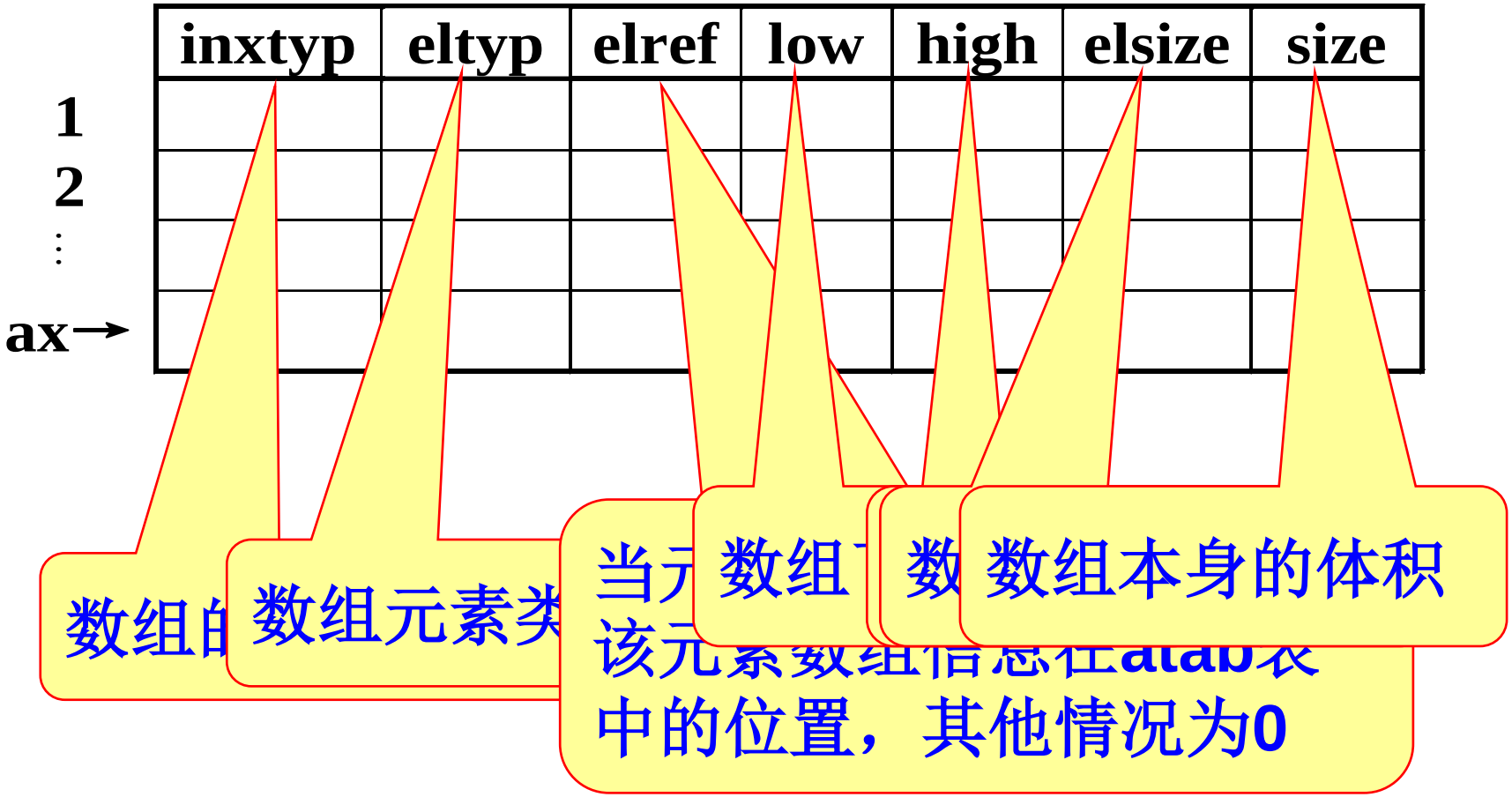



name
kind
lev
typ
normal
ref
adr/val/size
link
0
1
...
tx→
lastpar
last
psize
vsize
0
1
⋮
bx→
本程序体所有局部数据所需空间大小

# 层次显示表display：描述正在处理的各嵌套层，对程序体表进行管理



# (3)数组信息表atab


inxtyp
eltyp
elref
low
high
elsize
size
1
2
ax→
数组本身的体积
中的位置，其他情况为0

type a=array[1..10, 1..10] of integer;

| | name | kind | typ | ref | … |
|---|---|---|---|---|---|
| ⋮ | | | | | |
| k | a | type | arrays | n | |
| | | | | | |
| tx→ | | | | | |

nametab

| | inxtyp | eltyp | elref | low | high | elsize | size |
|---|---|---|---|---|---|---|---|
| ⋮ | | | | | | | |
| n | ints | arrays | m | 1 | 10 | 10 | 100 |
| ⋮ | | | | | | | |
| m | ints | ints | 0 | 1 | 10 | 1 | 10 |
| ax→ | | | | | | | |

atab

## (4) 中间代码表code

code：用于存放编译程序所产生的每条中间代码。