



编译原理

第八章 符号表

第八章 符号表

- 符号表的组织与作用
- 整理和查找
- 符号表的内容
- 名字的作用范围

8.4 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

PL 语言编译程序的符号表

■ 表格的定义

- 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)

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								

adr, 当名字为变量名时 (包括形参), 存入 (或形参) 在相应活动记录中分类的存储单元的相对地址; 对于过程名, 填入他们相应代码的相对地址

指向同一程序体中定义的上一个名字

val, 当名字为常量时, 存入常量的值; 当名字为过程名时, 存入该过程在程序体中的位置, 其他情况 ref

size, 当名字为数组名时, 存入数组的元素数目

link, 当名字为过程名时, 存入该过程在程序体中的位置, 其他情况 ref

	name	kind	lev	typ	normal	ref	adr/val/size	link
0								
1								
...								
tx→								

作用域进行分析；对名字表进行管理

	lastpar	last	psize	vsize
0				
1				
⋮				
bx→				

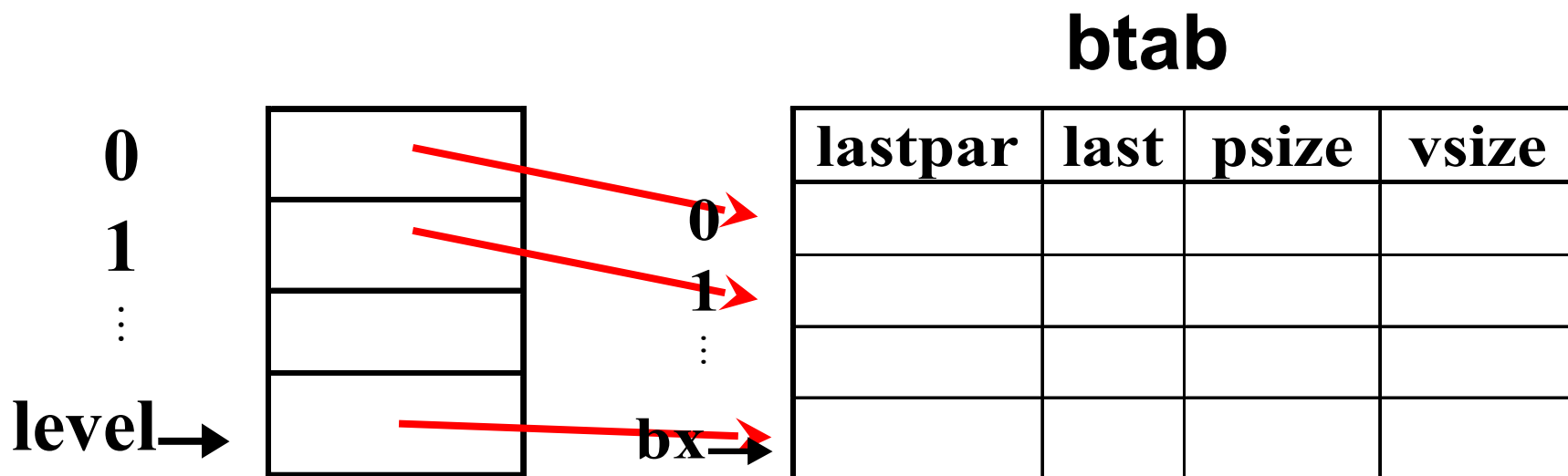
本程序体所有
包括连接数据

本程序体所有局部数据所
需空间大小

程序体表 btab 和层次显示表 display

■ 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



数组信息表 atab

- 用于记录每个数组的详细信息

	inxtyp	eltyp	elref	low	high	elsize	size
1							
2							
⋮							
ax →							

数组的

数组元素数

当为数组

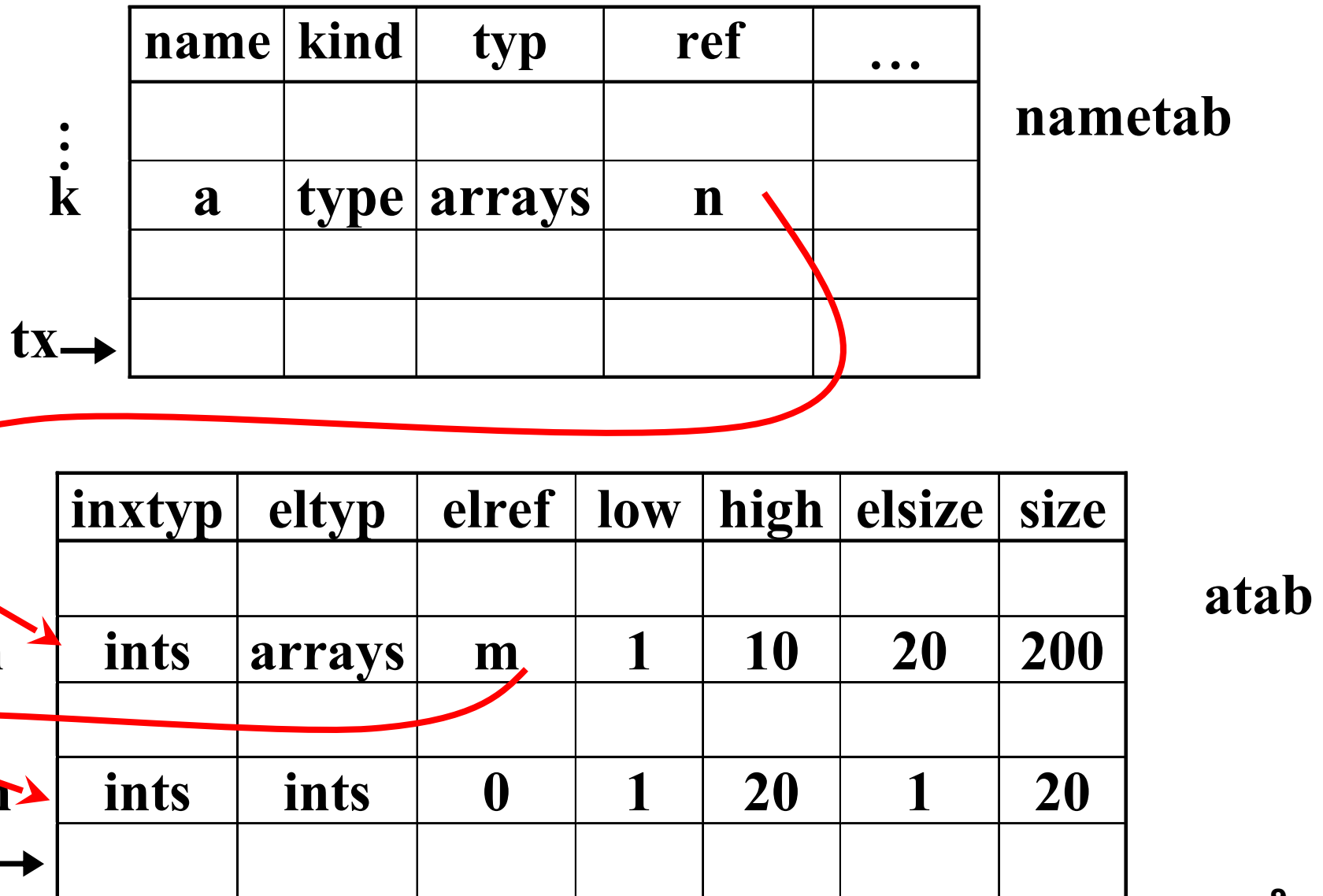
数

数组本身的体积

该元素数组信息在 atab
表中的位置，其他情况为

0

type a=array[1..10, 1..20] of integer;



中间代码表 code

- 用于存放编译程序所产生的每条中间代码

opcod	I	a

操作码

第一操作数，
程序体层数

第二操作数，
相对地址

第八章 符号表

- 符号表的组织与作用
- 整理和查找
- 符号表的内容
- 名字的作用范围

8.3 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
- 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过程中代表不同的名字
- 两种程序体结构
 - 并列结构，如 FORTRAN
 - 嵌套结构，如 PASCAL，ADA



```
PROGRAM  MAIN
```

```
...
```

```
integer X ,  Y
```

```
COMMON /C/A ,  B
```

```
...
```

```
END
```

```
SUBROUTINE SUB1
```


```
...
```

```
real X ,  Z
```

```
COMMON /C/A1 ,  B1
```

```
...
```

```
END
```



```
program P;  
  var a,b : integer;  
  procedure P1(i1, j1:integer);  
    var c,d : integer  
    ...  
  end;  
  procedure P2(i2, j2:integer);  
    var a,c : integer;  
    procedure P21;  
      var b1,b2 : boolean;  
      ...  
    end;  
  end;  
end.
```

符号表组织

■ FORTRAN 的符号表组织

- 一个 FORTRAN 程序由一个主程序段和若干个辅程序段组成
- 把局部名和全局名分别存在不同的地方

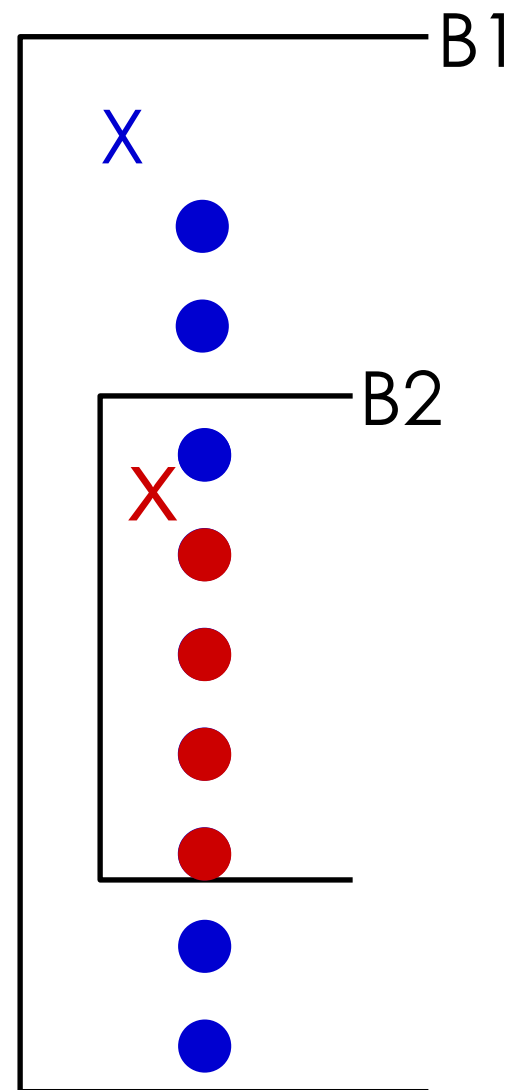
符号表组织

- FORTRAN 的符号表组织
- 嵌套结构语言的符号表组织
 - 如 PASCAL、ALGOL、ADA
- PASCAL 程序本身可以看成是一个操作系统所调用的过程，过程可以嵌套和递归
- 一个 PASCAL 过程

```
过程头;  
    说明段（由一系列的说明语句组成）;  
begin  
    执行体（由一系列的执行语句组成）;  
end
```


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



```

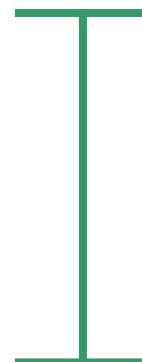
program main
  var A, B:real;
  ...
  procedure P1
    var B:boolean;
    ...
  begin
    ...
  end
  procedure P2
    var A:integer;
    ...
  begin
    ...
  end
begin
  ...
end

```

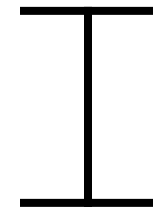
A(real)



A(integer)



B(real)



B(bool)



名字作用域分析

■ 两种做法

- 引入 " 过程编号 " 属性。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

To understand
a program you
must become
both the
machine and
the program.

PL 语言的中间代码

■ 指令格式: `opcod | a`

□ `opcod`: 操作码


□ `|`: 第一操作数, 程序体层数

□ `a`: 第二操作数, 相对地址

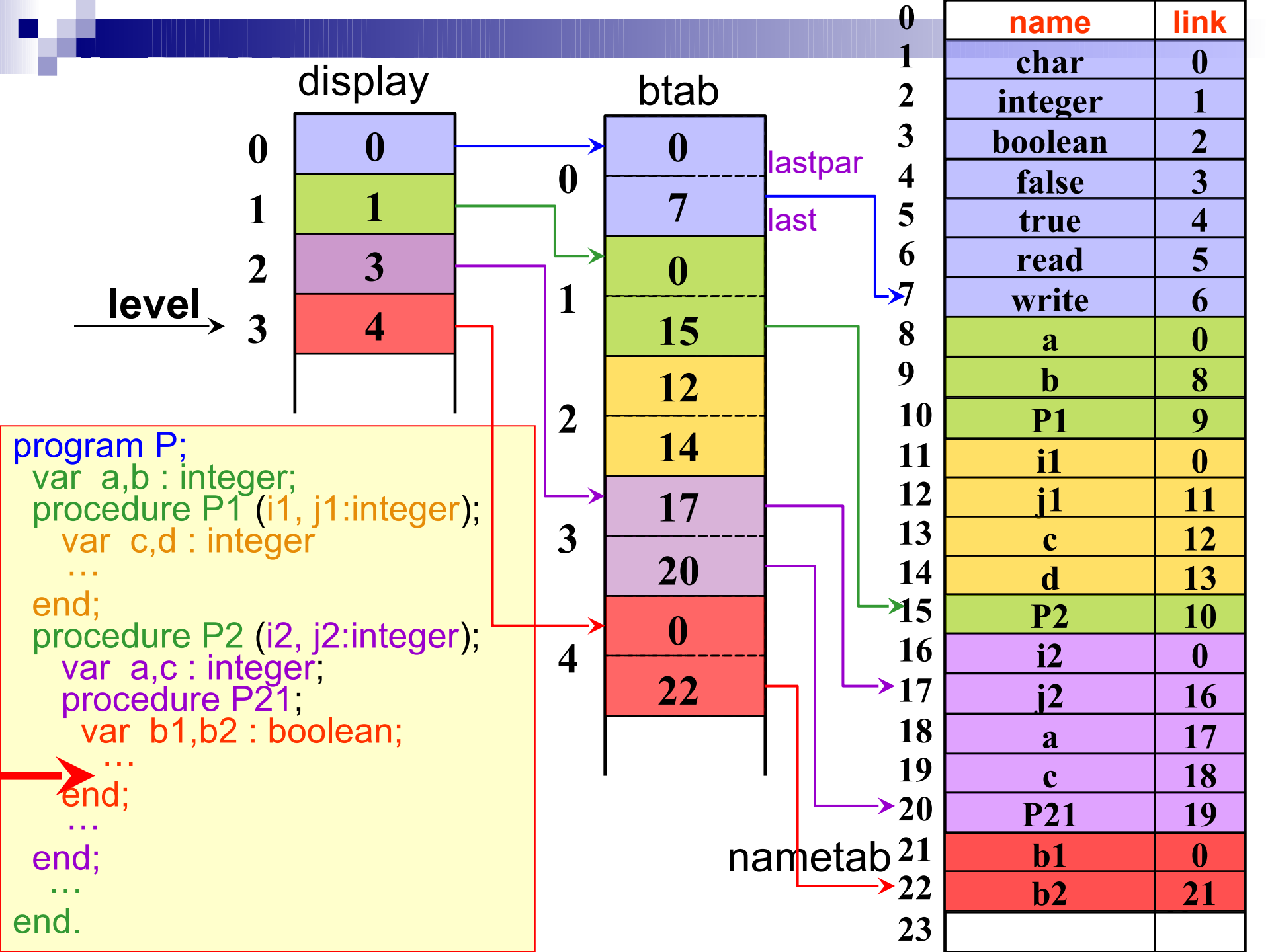
■ 编译时如何确定变量的层数 (地址)? Alan J. Perlis

■ 运行时如何根据指令中变量的层数和相对地址确定变量的存储单元?





```
program P;  
  var a,b : integer;  
  procedure P1 (i1, j1:integer);  
    var c,d : integer  
    ...  
  end;  
  procedure P2 (i2, j2:integer);  
    var a,c : integer;  
    procedure P21;  
      var b1,b2 : boolean;  
      ...  
    end;  
    ...  
  end;  
  ...  
end.
```



display

0	0
1	1
2	3
3	4

btab

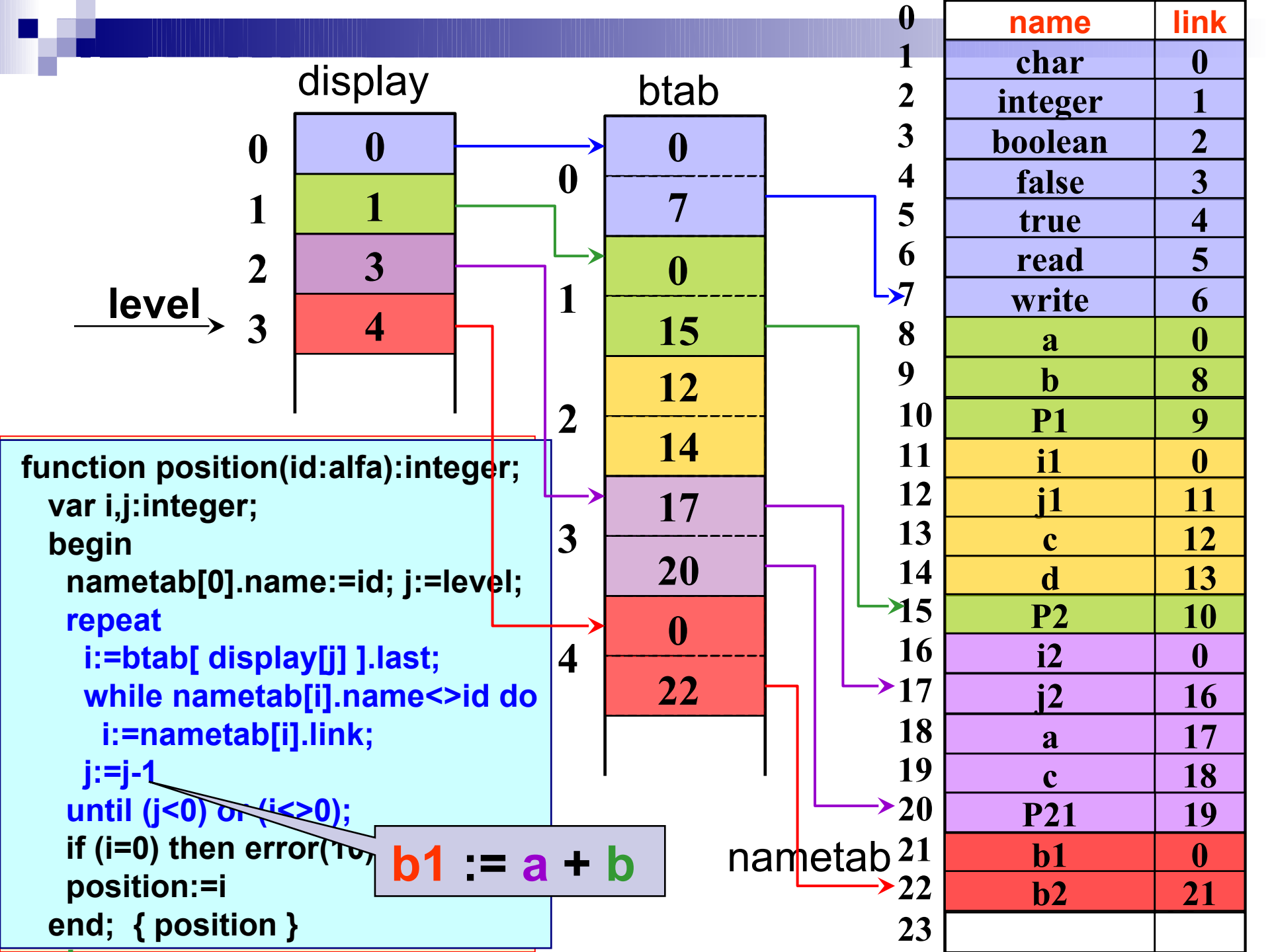
0	0
1	0
2	10
3	10
4	0

nametab

0	name	link
1	char	0
2	integer	1
3	boolean	2
4	false	3
5	true	4
6	read	5
7	write	6
8	a	0
9	b	8
10	P1	9
11	i1	0
12	j1	11
13	c	12
14	d	13
15	P2	10
16	i2	0
17	j2	16
18	a	17
19	c	18
20	P21	19
21	b1	0
22	b2	21
23		

```

program P;
var a,b : integer;
procedure P1 (i1, j1:integer);
    var c,d : integer
    ...
end;
procedure P2 (i2, j2:integer);
    var a,c : integer;
    procedure P21;
        var b1,b2 : boolean;
        ...
    end;
    ...
end.
    
```



PL 语言的中间代码

- 指令格式： `opcod I a`
 - `opcod`：操作码
 - `I`：第一操作数，程序体层数
 - `a`：第二操作数，相对地址
- 编译时如何确定变量的层数（地址）？
- 运行时如何根据指令中变量的层数和相对地址确定变量的存储单元？

本章小结

- 符号表的组织与作用
- 整理和查找
- 符号表的内容
- 名字的作用范围

作业

- 《编译原理大作业实习》

- 阅读

- 第三节 PL 语法分析
 - 分析 PL 编译程序的数据结构及相关程序

- 预习

- 第四节 PL 语义分析及中间代码产生