编译原理

--符号表

刘 爽

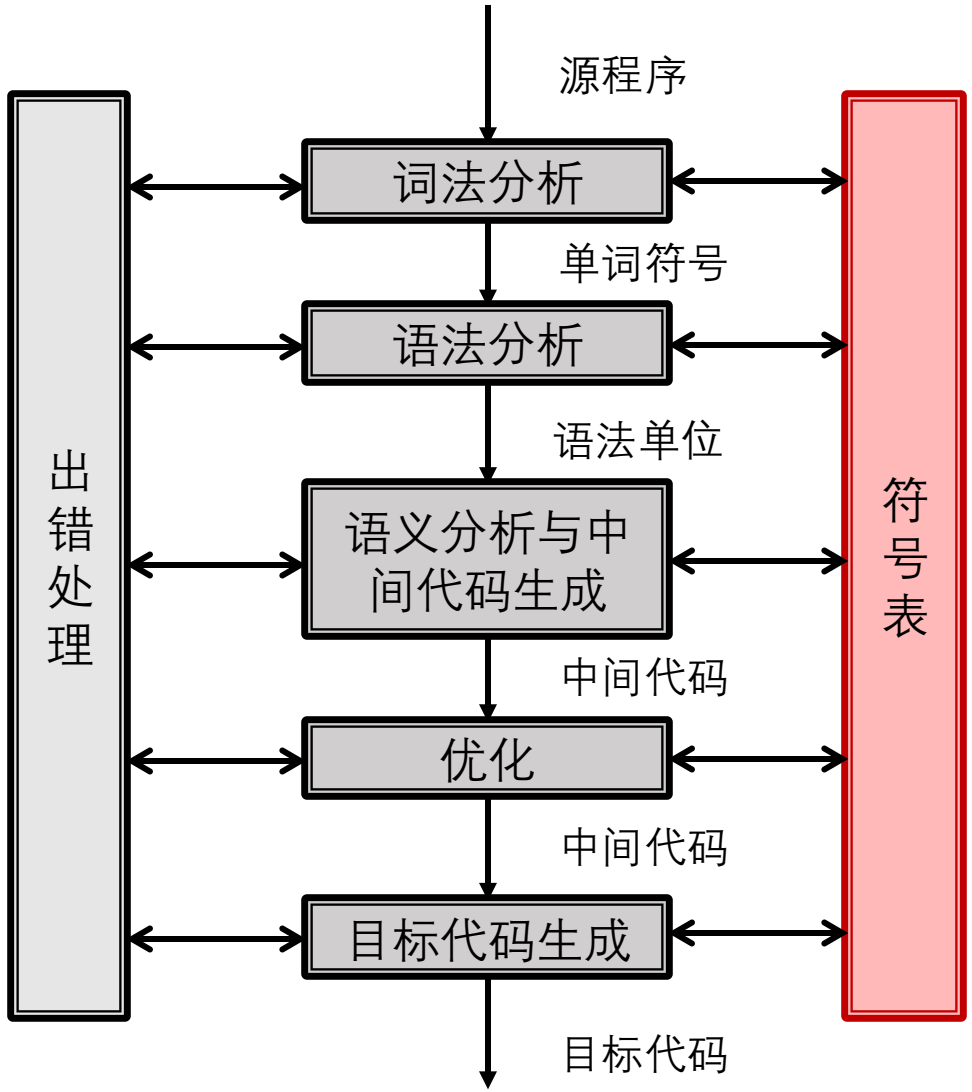
中国人民大学信息学院

内容提要

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

符号表组织与作用

- 在编译程序工作的过程中，需要不断汇集和反复查证出现在源程序中的各种名字的属性和特征等有关信息。这些信息记录在一张或几张符号表中。
  - 用于静态语义检查和产生中间代码
  - 目标代码生成阶段，符号表是对符号进行地址分配的依据
  - 用来体现作用域与可见性信息
- 符号表的组织、构造和管理方法的好坏直接影响编译系统的运行效率。
- 对一个多遍扫描的编译程序，不同遍所用的符号表也会有所不同，因为每遍所关心的信息或所能得到的信息会有差异



符号表的组织与作用

- 符号表中的每一项（入口）包含两部分：
  - 名字栏，也称主栏，关键字栏，如程序名、过程名、函数名、变量名等
  - 信息栏，记录相应的不同属性，如类型、值、地址、作用域、其他信息（数组内情向量、记录结构的成员信息、函数及过程的形参等）

	名字栏（Name）	信息栏（Information）
第1项（入口1）		
	…	
第n项（入口n）		

符号表的组织与作用

- 符号表的基本操作大致可以归纳为六大类：

  - **创建符号表：在编译开始，或进入一个作用域**
  - **插入表项：在遇到新的标识符声明时进行**
  - **查询表项：在引用标识符时进行**
  - **修改表项：在获得新的语义值信息时进行**
  - **删除表项：在标识符成为不可见或不再需要它的任何信息时进行**
  - **释放符号表空间：在编译结束前或退出一个作用域时**

名字栏（Name）	信息栏（Information）
sum	实型，变量
index	整型，变量
str	数组，形参

- 不同种类的表格所涉及的操作也会存在不同，上述是基本操作。

符号表的组织与作用

- 符号表的作用
  - 一致性检查和作用域分析
  - 辅助代码生成

- 对符号表进行操作的时机
  - 定义性出现

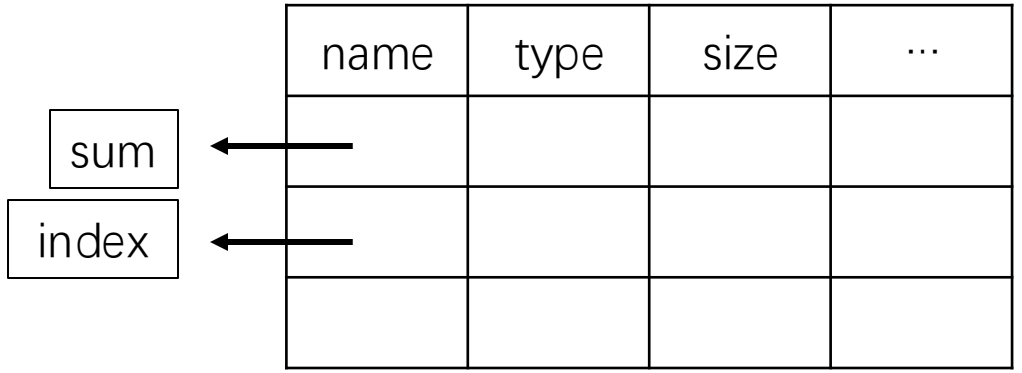
    如： int index; double sum = 0;
  - 使用性出现

    如： if (index > 0) { }

符号表的组织与作用

- 符号表的组织方式
  - 安排各项各栏的存储单元为固定长度：组织简单，易造成空间浪费
  - 用间接方式安排各栏存储单元：灵活性强，相对复杂

name	type	size	…
sum			
index			
…			



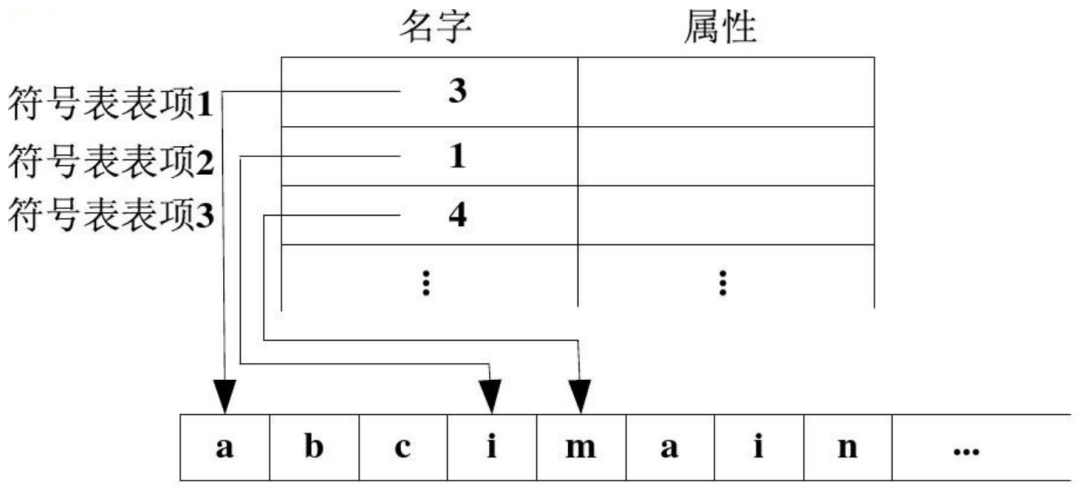
符号表的组织与作用

- 符号表的组织方式
  - 安排各项各栏的存储单元为固定长度：组织简单，易造成空间浪费
  - 用间接方式安排各栏存储单元：灵活性强，相对复杂

存储单元为固定长度

名字	属性
abc	
i	
main	
…	

用间接方式安排各栏存储单元



(a)标识符长度放在符号表中

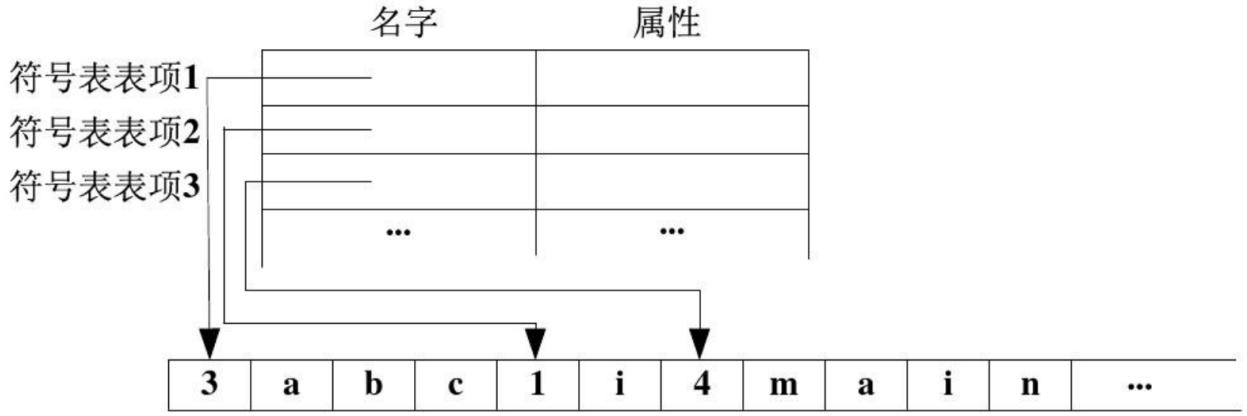
符号表的组织与作用

- 符号表的组织方式
  - 安排各项各栏的存储单元为固定长度：组织简单，易造成空间浪费
  - 用间接方式安排各栏存储单元：灵活性强，相对复杂

存储单元为固定长度

名字	属性
abc	
i	
main	
…	

用间接方式安排各栏存储单元



(*b*) 标识符长度放在字符串中

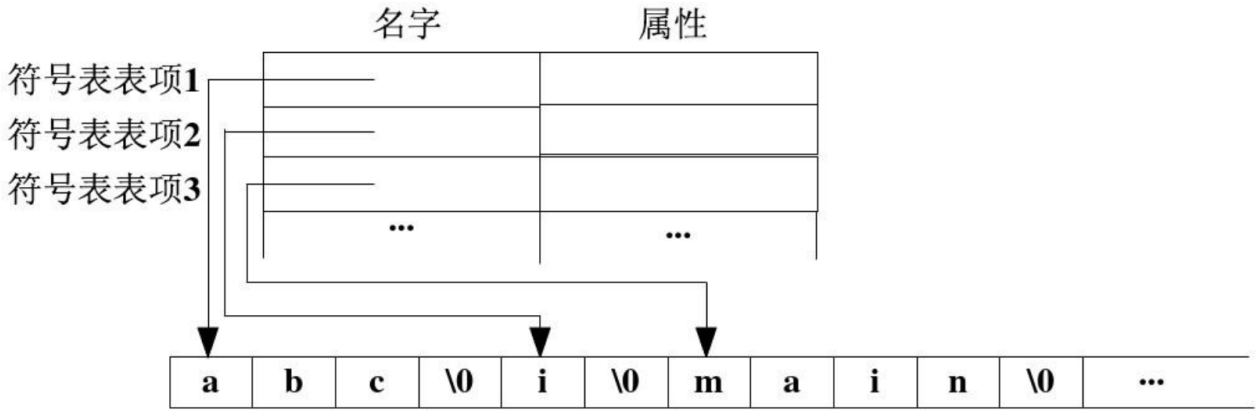
符号表的组织与作用

- 符号表的组织方式
  - 安排各项各栏的存储单元为固定长度：组织简单，易造成空间浪费
  - 用间接方式安排各栏存储单元：灵活性强，相对复杂

存储单元为固定长度

名字	属性
abc	
i	
main	
…	

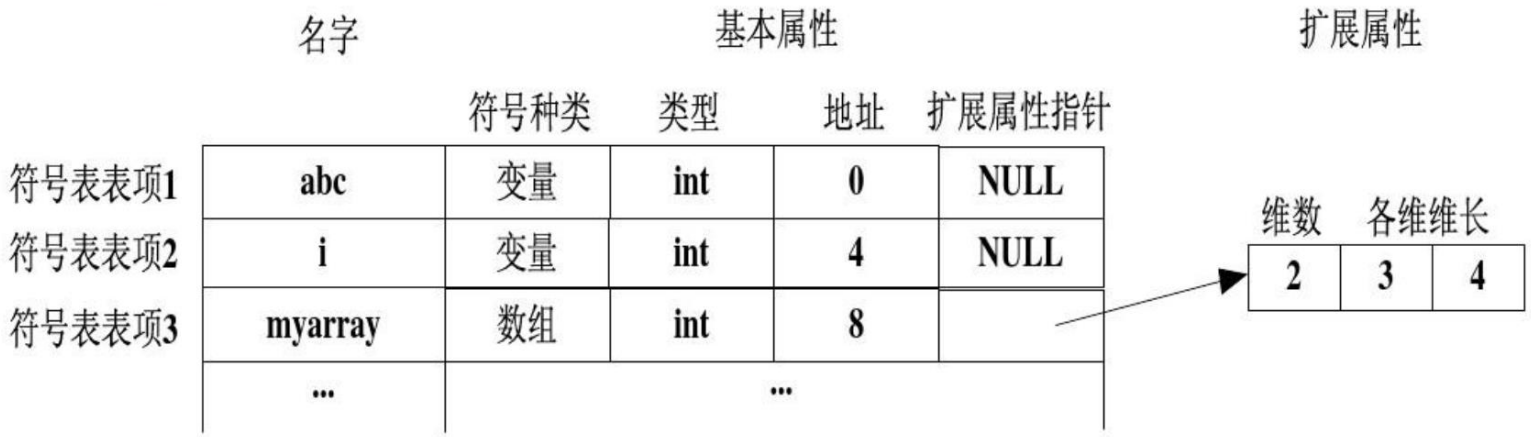
用间接方式安排各栏存储单元



(c) 用’\0’表示标识符的结束

符号表的组织与作用

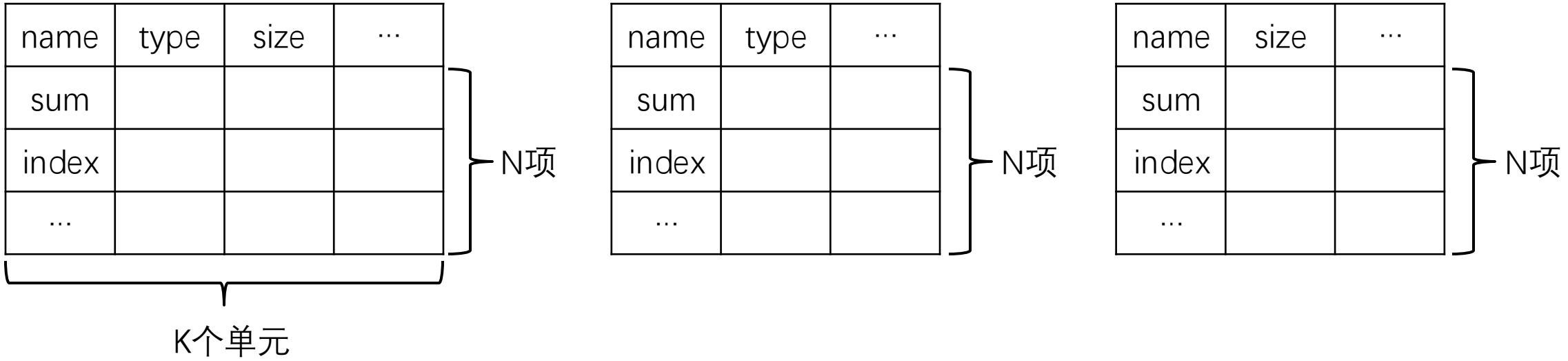
- 通过符号表访问内情向量
  - 不同符号所需信息空间长度不一样，可把共同属性放在符号表中，而把特殊属性存放在别的地方，在符号表中设一个指针指示信息存放的位置。



- 可以按名字的不同种属建立多张符号表，如常数表、变量名表、过程名表等

符号表的组织与作用

- 符号表的存放次序
  - 把每一项置于连续的K个存储单元中，构成一张K*N个字的表
  - 把整个符号表分成M个子表，如T_1，T_2，…，T_m，每个子表含N项



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.
```

符号名表SNT

	NAME	INFORMATION
(1)	M	形式参数，整型，变量
(2)	N	形式参数，整型，变量
(3)	K	整型，变量

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.
```

常数表CT

	值（VALUE）
(1)	1
(2)	4

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.
```

入口名表ENT

	NAME	INFORMATION
(1)	INCWAP	二目子程序，入口QT（1）

四元式表QT

	OPR	ARG1	ARG2	RESULT
(1)	LINK			
(2)	actpar	INCWAP	1	M
(3)	actpar	INCWAP	2	N
(4)	+	M	1	K
(5)	+	N	4	M
(6)	:=	K		N
(7)	return			

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.
```

标号表

	NAME	INFORMATION
(1)	START	QT（4）

四元式表QT

	OPR	ARG1	ARG2	RESULT
(1)	LINK			
(2)	actpar	INCWAP	1	M
(3)	actpar	INCWAP	2	N
(4)	+	M	1	K
(5)	+	N	4	M
(6)	:=	K		N
(7)	return			

符号表的实现

– 实现符号表的常用数据结构

- 一般的线性表

  如：数组，链表等

- 有序表

  查询较无序表快，如可以采用折半查找

- 二叉搜索树

- Hash表

– 存储效率问题

- 重要，但本课程不专门讨论
- 两方面：省空间，高效率

内容提要

- 符号表的组织与作用
- 整理与查找
  - 线性表
  - 对折查找与二叉树
  - 杂凑技术（HASH技术）
- 名字的作用范围
- 符号表的内容

线性查找

- 按关键字出现的顺序填写各项
  - 填表快，查找慢
  - 结构简单，节省空间，效率低，查找时间复杂度：O(n)
  - 改进：自适应线性表，如最新最近使用

例如：

```
int a;
int b;
if(a > b) {
   int d = a - b;
   return d;
} else {
   int c = b – a;
   return c;
}
```

起始地址：h→
（查找起始）

结束地址：p→
（插入起始）

NAME	INFORMATION
a	
b	
d	
c	

对折查找与二叉树

- 表格中的项按名字的“大小”顺序整理排列
  - 填表慢，查找快
  - 查找时间复杂度：$O(Log_2n)$
  - 改进：组织成二叉树

插入：根据排序选择插入位置
查找：根据大小关系二分查找

例如：

```
int a;
int b;
if(a > b) {
   int d = a - b;
   return d;
} else {
   int c = b – a;
   return c;
}
```

起始地址：h→

结束地址：p→

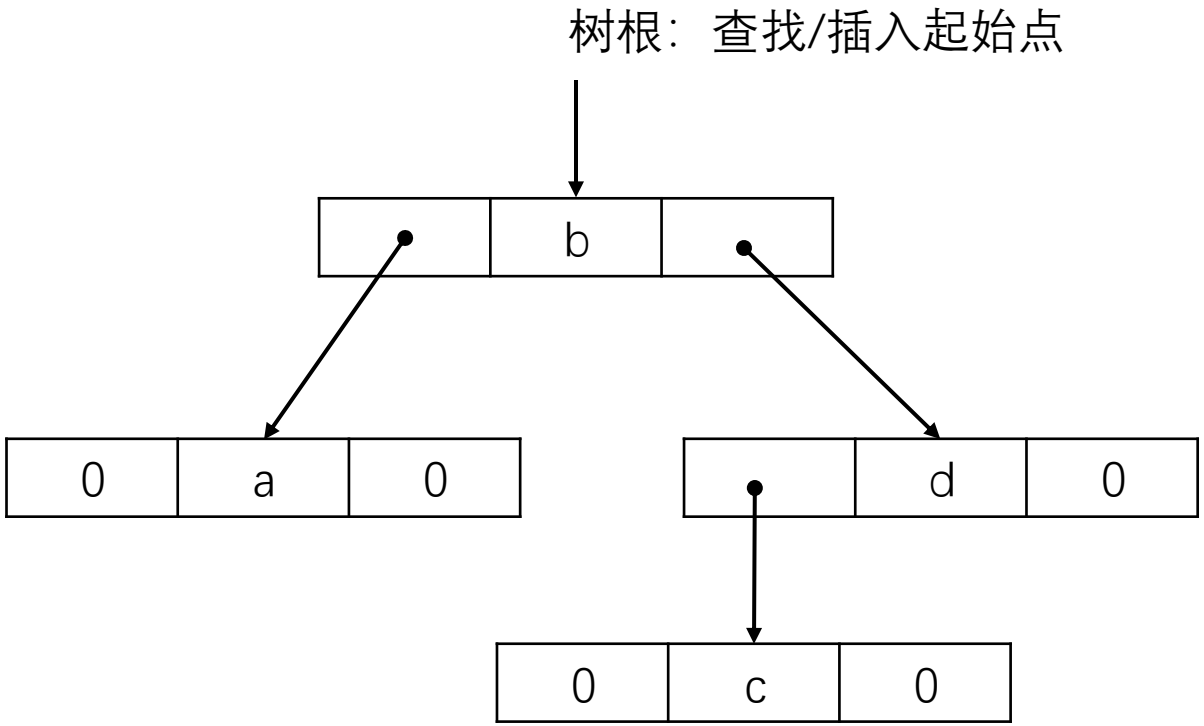
NAME	INFORMATION
a	
b	
c	
d	

对折查找与二叉树

- 表格中的项按名字的“大小”顺序整理排列
  - 填表慢，查找快
  - 查找时间复杂度：$O(Log_2n)$
  - 改进：组织成二叉树

例如：

```
int a;
int b;
if(a > b) {
   int d = a - b;
   return d;
} else {
   int c = b – a;
   return c;
}
```



杂凑技术

- 一种争取查表、填表两方面都能高效进行的技术。
  - 这种办法是：假定有一个足够大的区域，这个区域是以填写一张含N项的符号表。我们希望构造一个地址函数 H，对任何名字 SYM，H(SYM) 取值于0至N-1之间。这就是说，不论对 SYM 查表或填表，我们都希望能从 H(SYM) 获得它的表中的位置。
  - 例如，我们用无符号整数作为项名，N=17。把 H(SYM) 定义为SYM%N。
    - 名字“09”将被置于表中的第9项；“34”置于第0项；“171” 置于第1项。
- 地址函数 H 要求
  - 计算简单、高效
  - 函数值分布均匀

优点：填表快，查找快：O(1)
缺点：地址冲突 （参考数据结构内容）

内容提要

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

名字的作用范围

- 在程序语言中，名字都有一个确定的能被使用的区域范围，被称为这个名字的作用域。
- 允许同一个标识符在不同的过程中代表不同的名字。

- 两种程序体结构：
  - 并列结构，如 FORTRAN
  - 嵌套结构，如PASCAL

FORTRAN示例

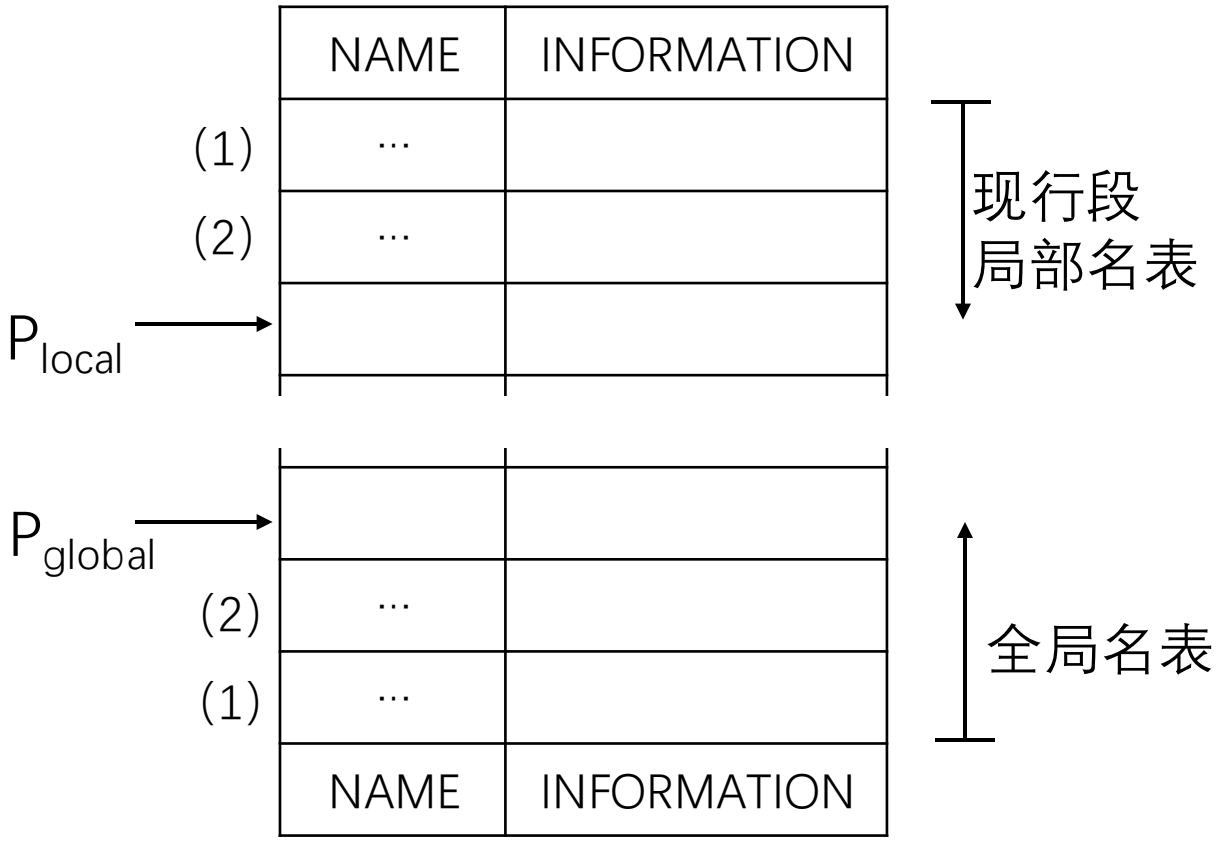
分析不同程序段时
更新局部名表

```
PROGAM MAIN
  …
integer X, Y
COMMON /C/A,B
  …
END
```

程序段 1

```
SUBROUTINE SUB1
  …
real X, Z
COMMON C/A1, B1
  …
END
```

程序段 2



符号表的作用域

- 作用域与可见性
  - 嵌套的作用域（*nested scopes*）
  - 开作用域与闭作用域（相应于程序中特殊点）
    - 该点所在的作用域为当前作用域
    - 当前作用域与包含它的程序单元所构成的作用域称为开作用域（*open scopes*）
    - 不属于开作用域的作用域称为闭作用域（*close scopes*）

常用的可见性规则（*visibility rules*）

- 在程序的任何一点，只有在该点的开作用域中声明的名字才是可访问的
- 若一个名字在多个开作用域中被声明，则把离该名字的某个引用最近的声明作为该引用的解释
- 新的声明只能出现在当前作用域

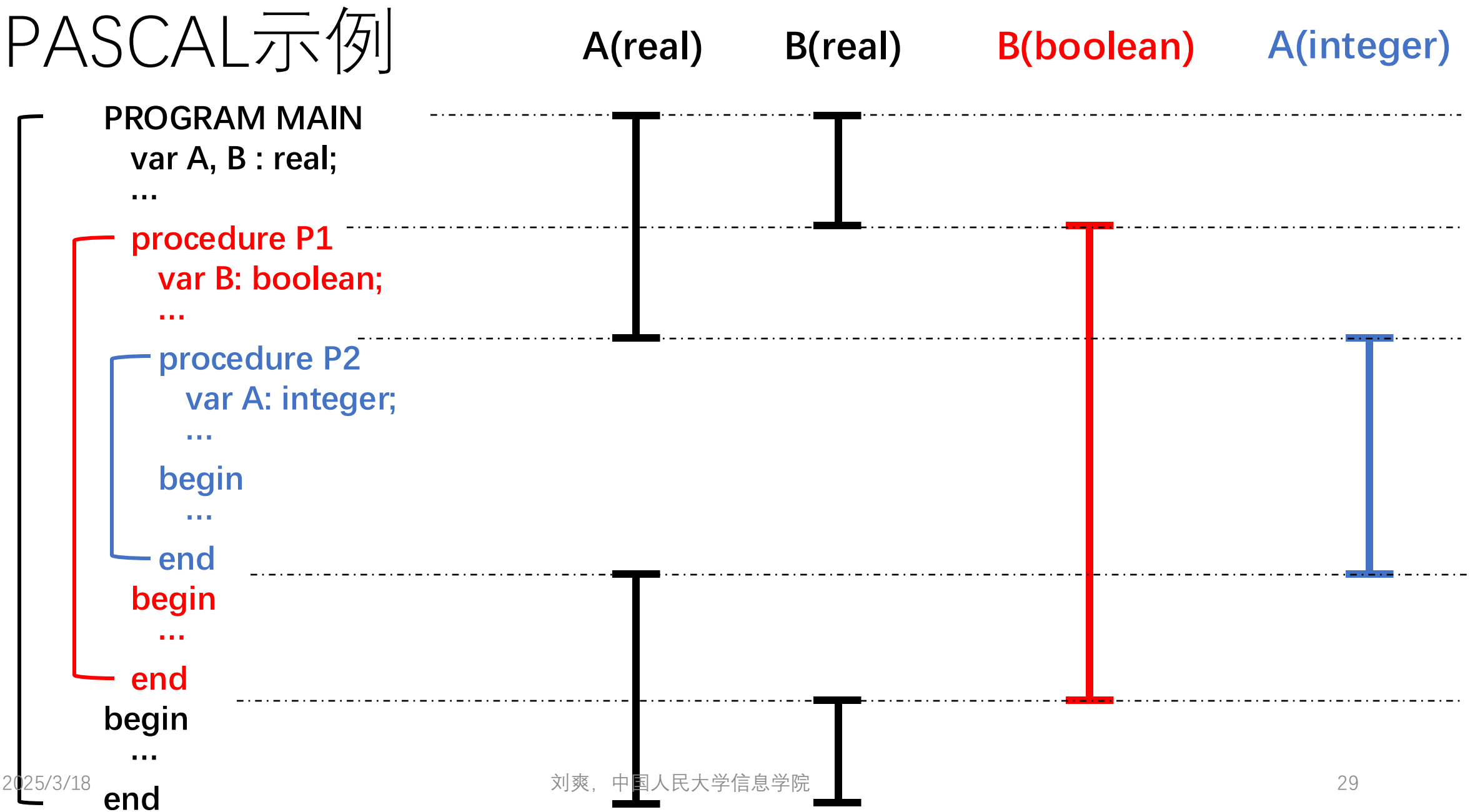
符号表的作用域

- 作用域与单符号表组织
  - 所有作用域共用一个全局符号表
  - 每个作用域都有各自的符号表
  - 仅记录开作用域中的符号
  - 当某个作用域成为闭作用域时，从符号表中删除该作用域中所声明的名字
- 作用域与多符号表组织
  - 每个作用域都有各自的符号表
  - 维护一个符号表的作用域栈，每个开作用域对应栈中的一个入口，当前的开作用域出现在该栈的栈顶
  - 当一个新的作用域开放时，新符号表将被创建，并将其入栈
  - 在当前作用域成为闭作用域时，从栈顶弹出相应的号表

作用域嵌套

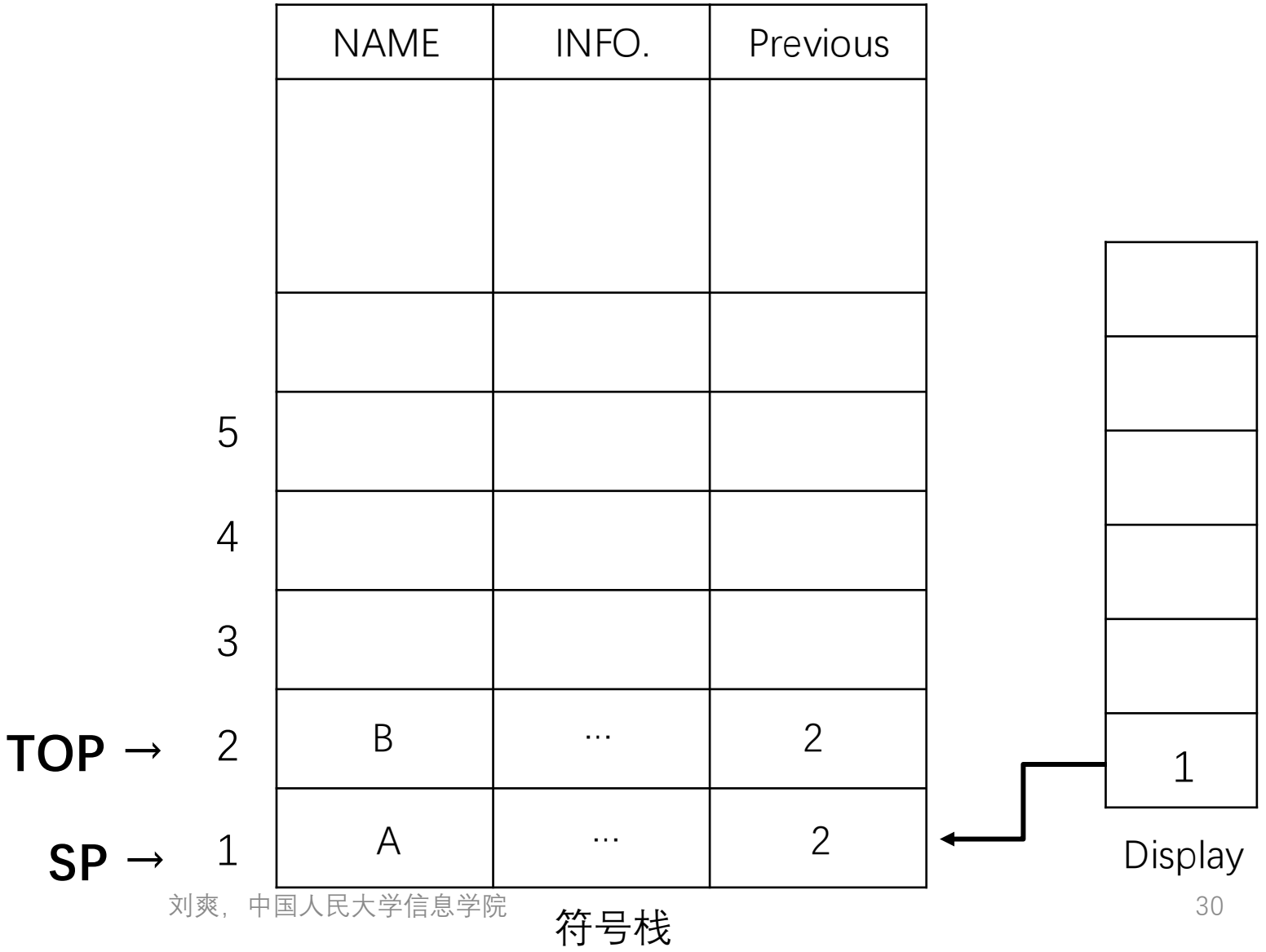
- 嵌套结构作用域规则 —— “最近嵌套原则”
  - 一个在子程序 B1 中说明的名字 X，只在 B1 中有效（局部于B1）
  - 如果 B2 是 B1 的一个内层子程序，且 B2 中对标识符 X 没有新的说明，则原来的名字 X 在 B2 中仍然有效。如果 B2 对 X 重新做了说明，那么，B2 对 X 的任何引用都是指重新说明过的 X。

PASCAL示例


A(real)
B(real)
B(boolean)
A(integer)
PROGRAM MAIN
var A, B : real;
...
procedure P1
var B: boolean;
...
procedure P2
var A: integer;
...
begin
...
end
begin
...
end
begin
...
end

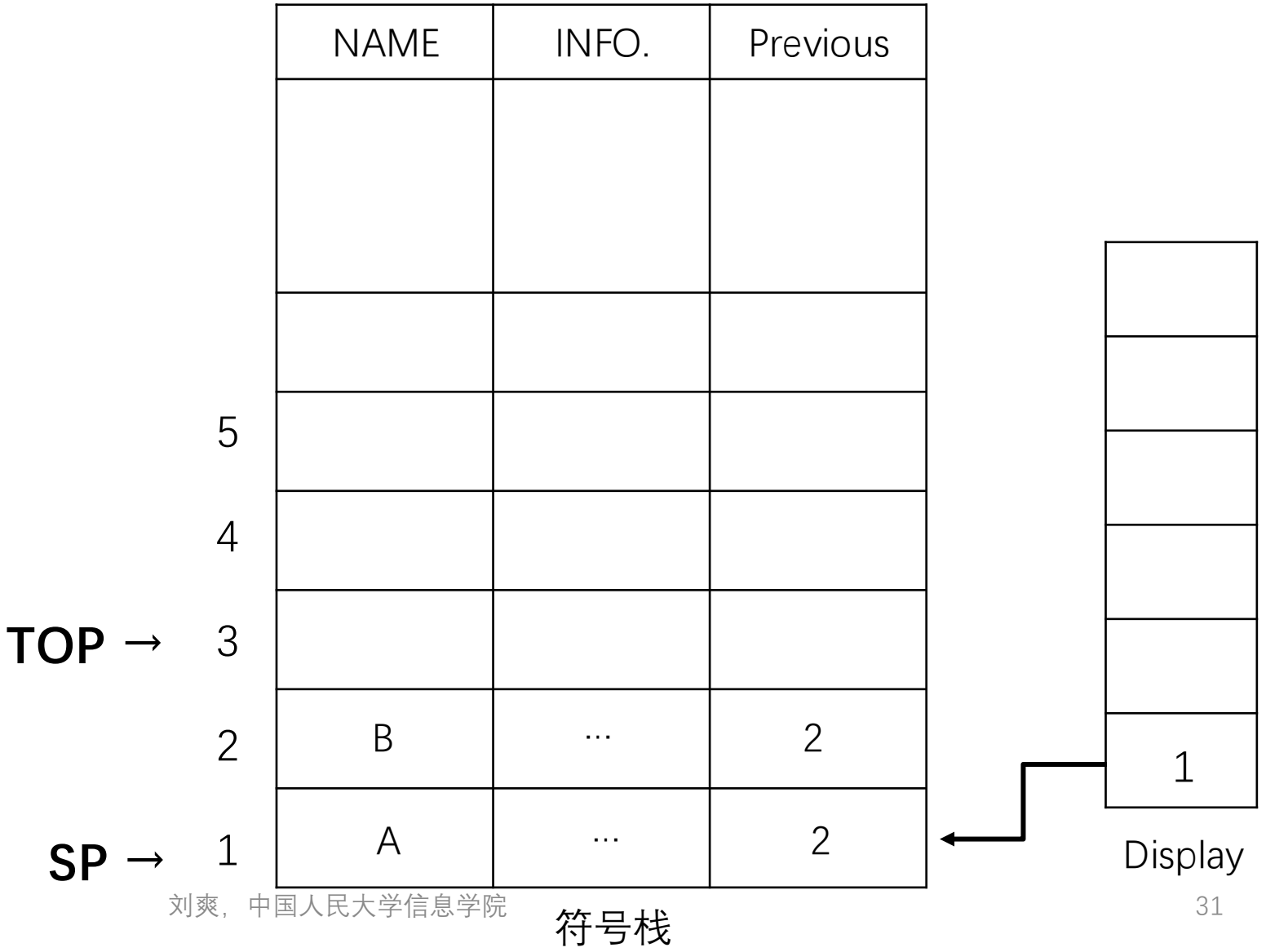
作用域嵌套栈式存储

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



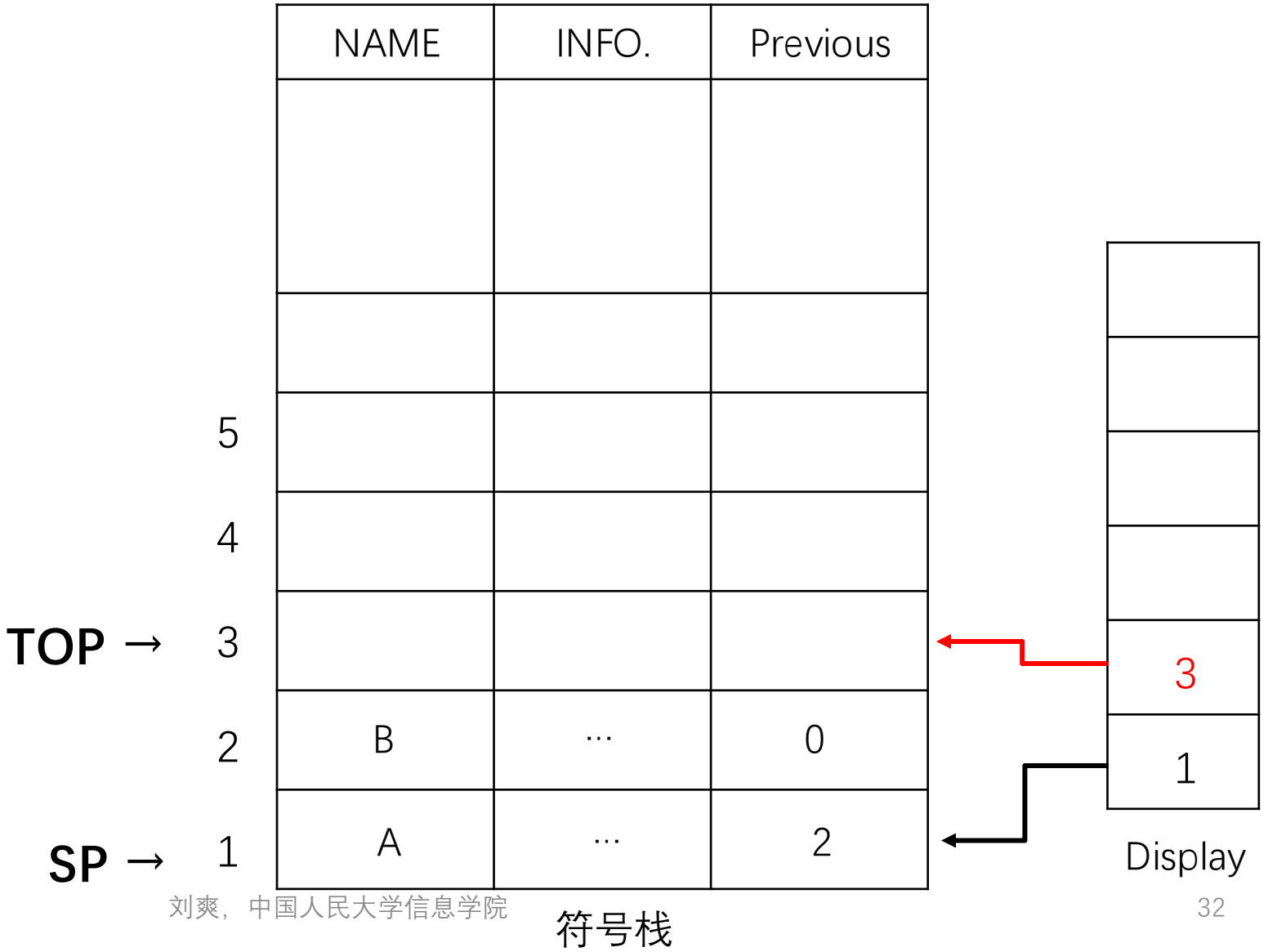
作用域嵌套栈式存储

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



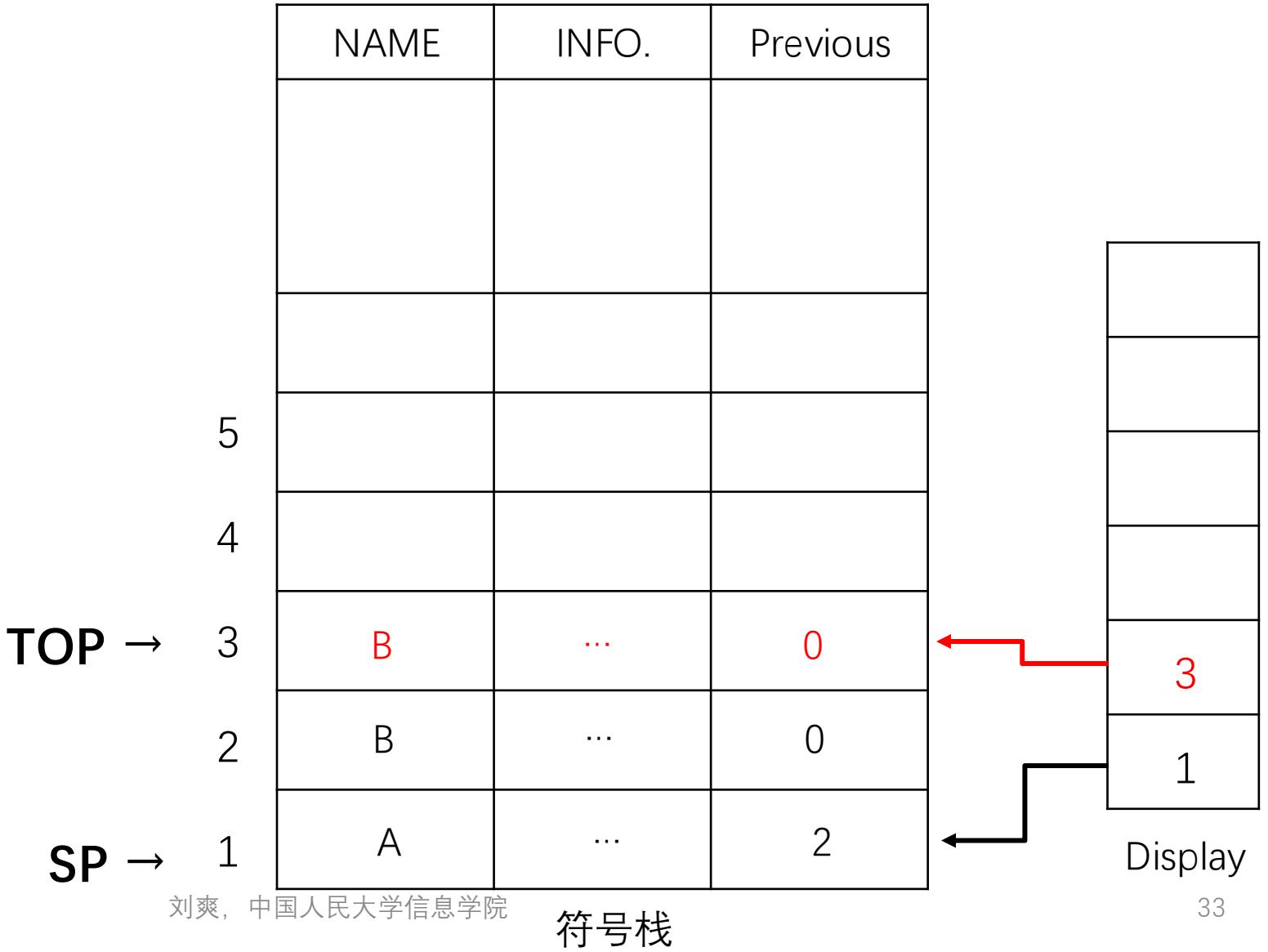
作用域嵌套栈式存储

```
PROGRAM MAIN
  var A, B : real;
  ...
  procedure P1
    var B: boolean;
    ...
    procedure P2
      var A: integer;
      ...
    begin
      ...
    end
  begin
    ...
  end
begin
  ...
end
```



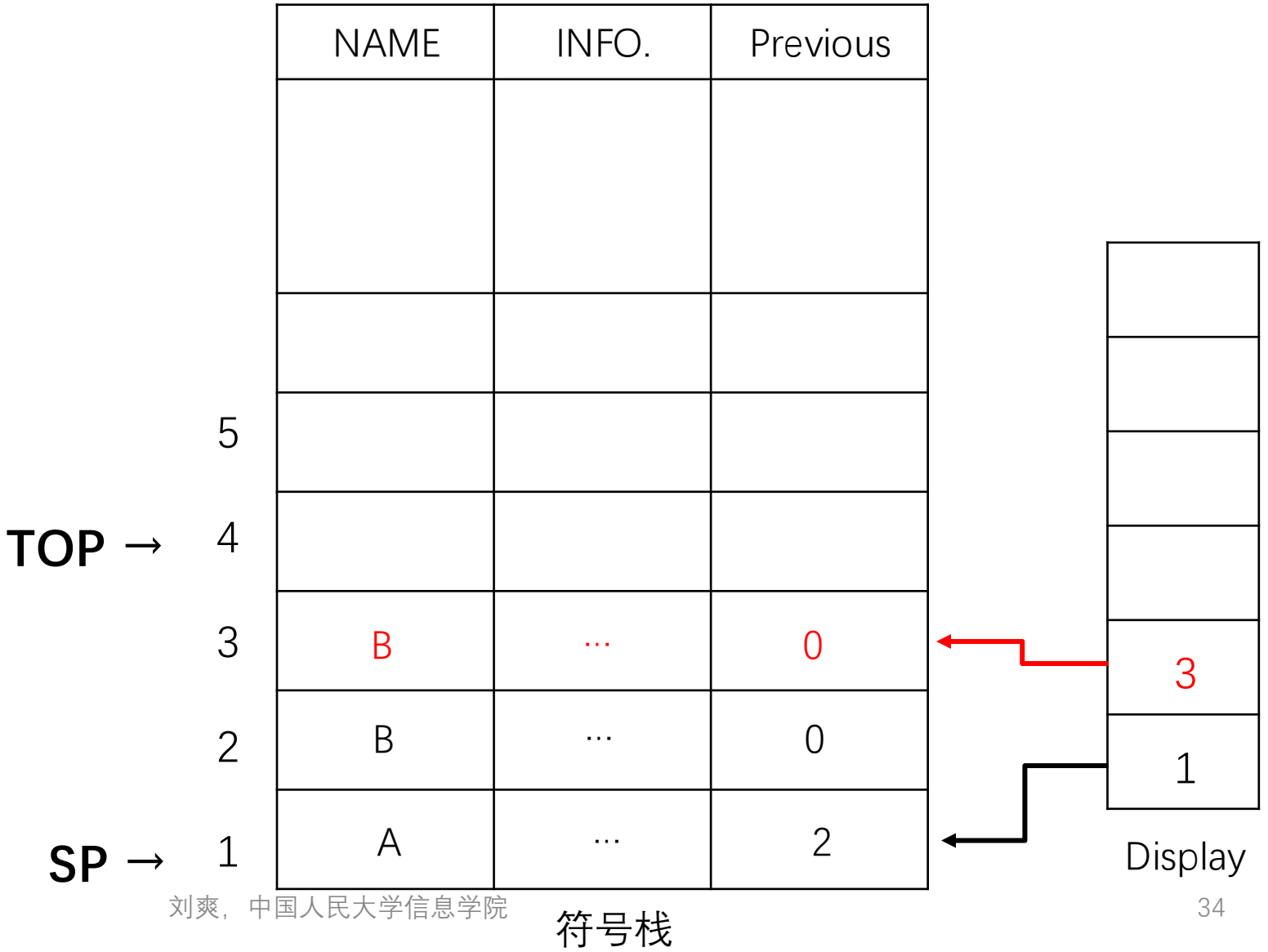
作用域嵌套栈式存储

```
PROGRAM MAIN
  var A, B : real;
  ...
  procedure P1
    var B: boolean;
    ...
    procedure P2
      var A: integer;
      ...
    begin
      ...
    end
  begin
    ...
  end
begin
  ...
end
```



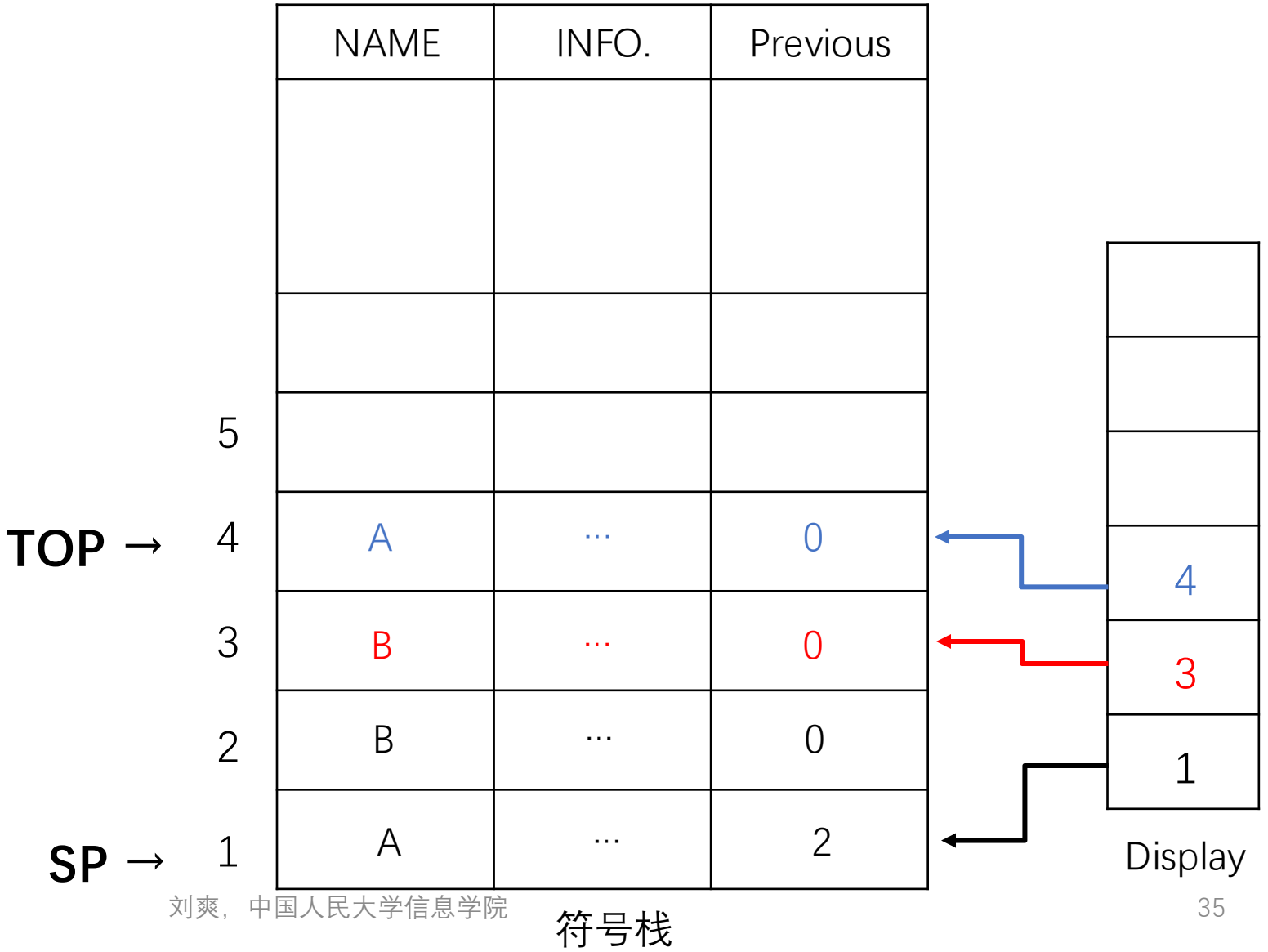
作用域嵌套栈式存储

```
PROGRAM MAIN
  var A, B : real;
  ...
  procedure P1
    var B: boolean;
    ...
    procedure P2
      var A: integer;
      ...
    begin
      ...
    end
  begin
    ...
  end
begin
  ...
end
```



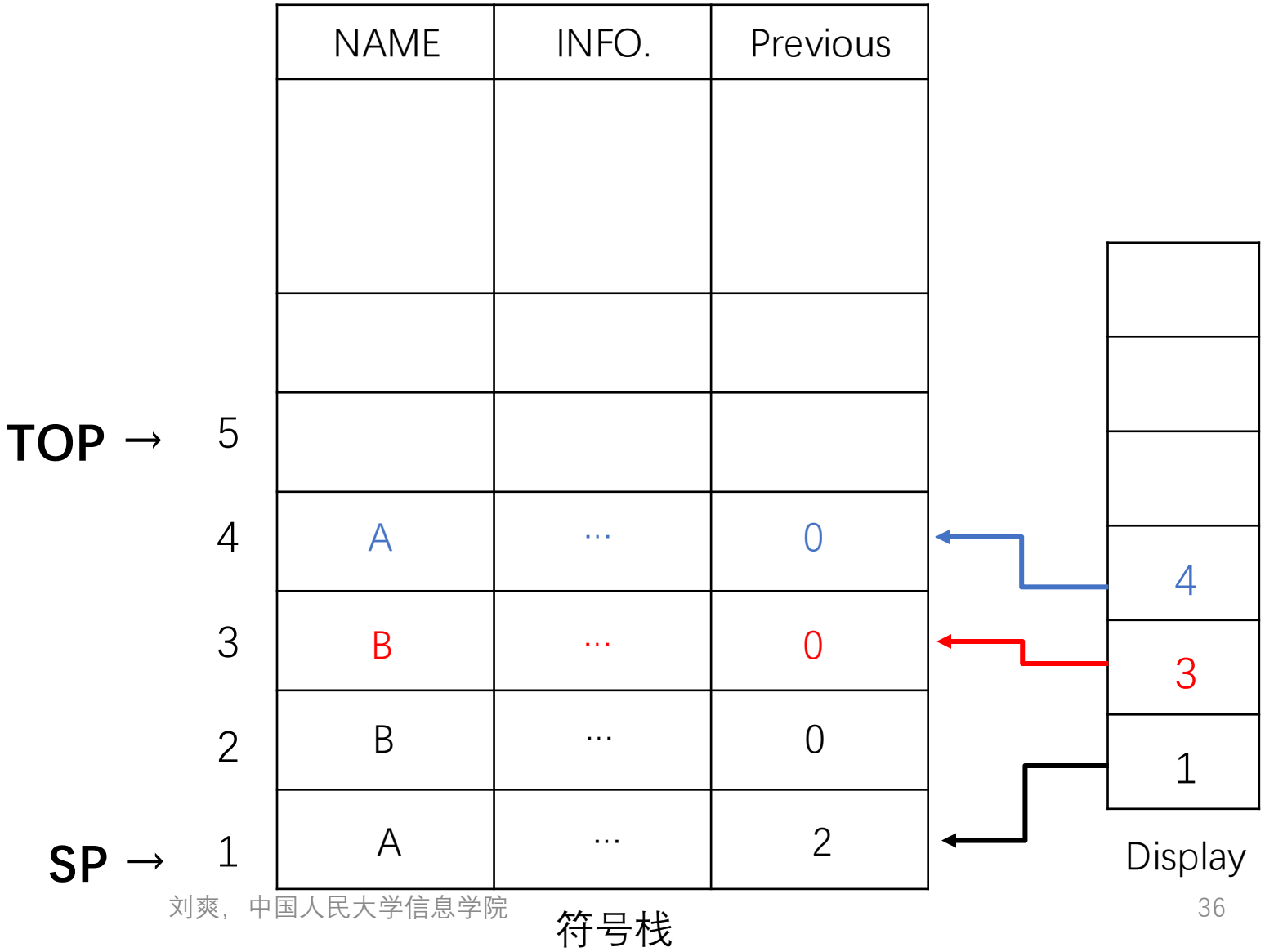
作用域嵌套栈式存储

```
PROGRAM MAIN
  var A, B : real;
  ...
  procedure P1
    var B: boolean;
    ...
    procedure P2
      var A: integer;
      ...
    begin
      ...
    end
  begin
    ...
  end
begin
  ...
end
```



作用域嵌套栈式存储

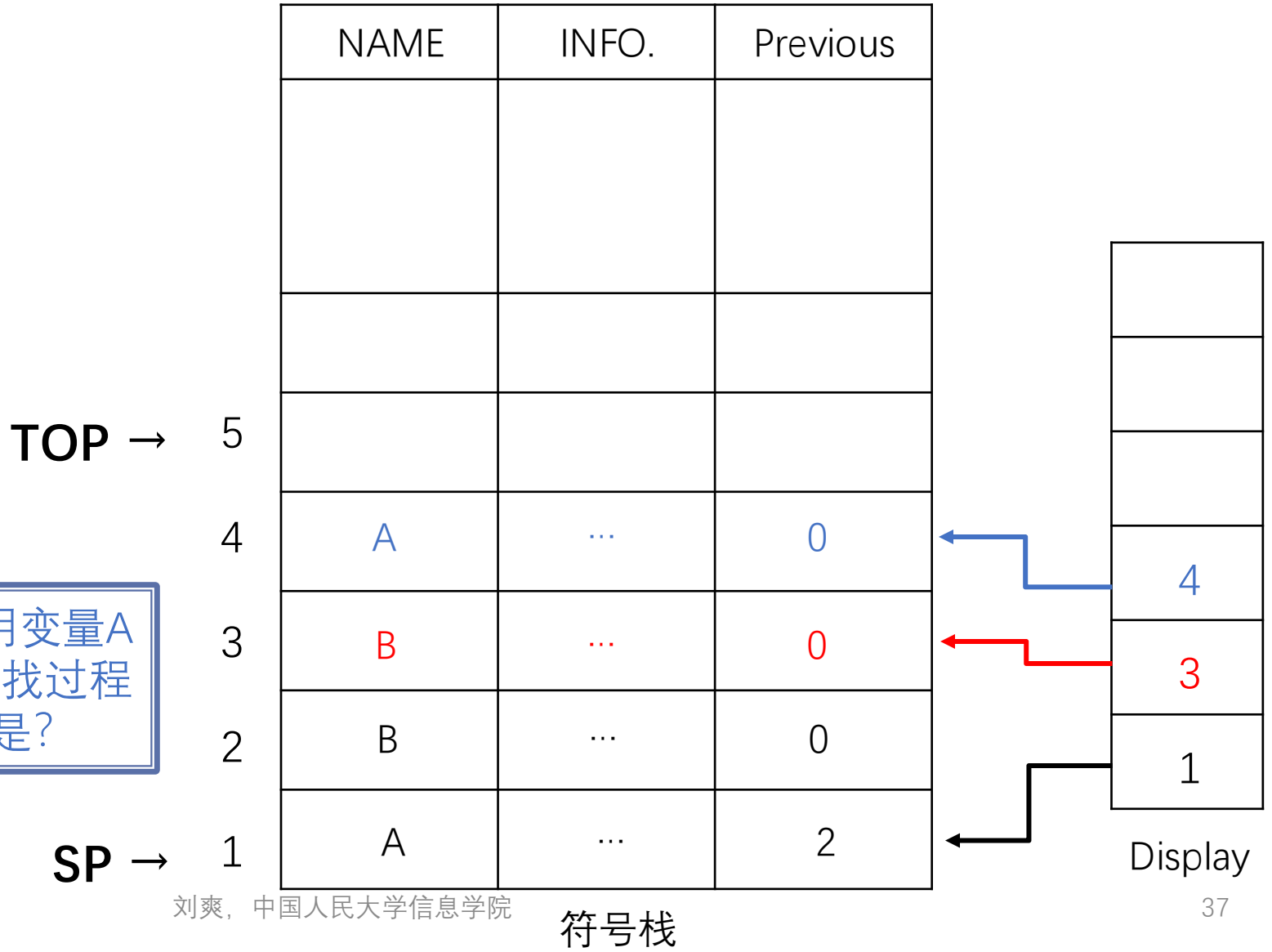
```
PROGRAM MAIN
  var A, B : real;
  ...
  procedure P1
    var B: boolean;
    ...
    procedure P2
      var A: integer;
      ...
    begin
      ...
    end
  begin
    ...
  end
begin
  ...
end
```



作用域嵌套栈式存储

```
PROGRAM MAIN
  var A, B : real;
  ...
  procedure P1
    var B: boolean;
    ...
    procedure P2
      var A: integer;
      ...
    begin
      ...
    end
  begin
    ...
  end
begin
  ...
end
```

此处引用变量A和B，查找过程分别是？



内容提要

- 符号表的组织与作用
- 整理与查找
- 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，如类型、种属、大小以及相对数等。不同的语言对名字性质的定义各不相同。

- 对于变量名、数组名和过程名而言，一般要有下列信息：
  - 变量——类型（整型、实型、指针等）、种属（简单变量、数组等）、长度（所需的存储单元数）、相对数（存储单元的相对地址）
  - 数组——记录内情向量
  - 记录——把它与其分量按某种形式联系起来
  - 形参标志
  - 是否赋值的标志位等

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，如类型、种属、大小以及相对数等。不同的语言对名字性质的定义各不相同。

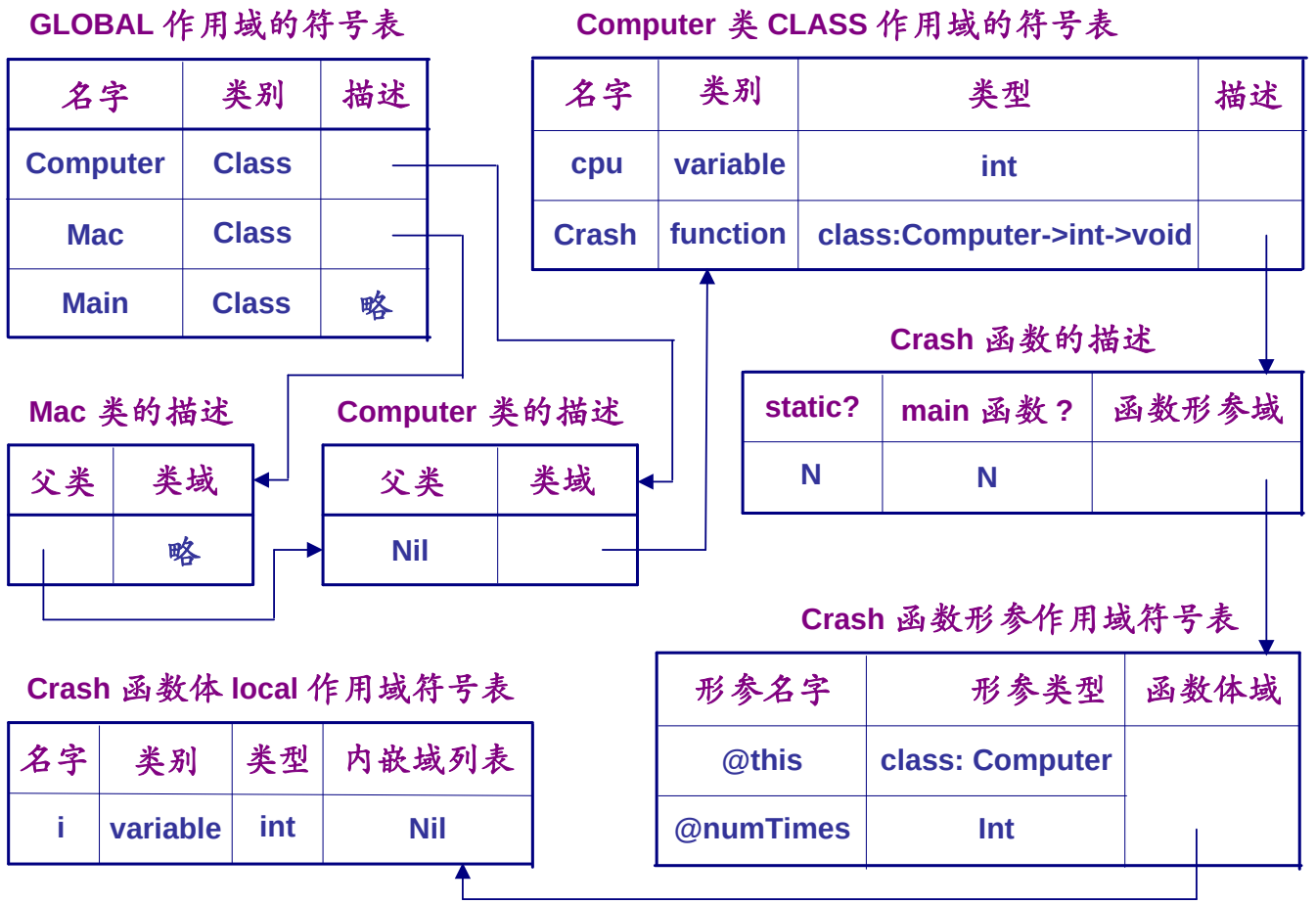
- 对过程，一般要有下列信息：
  - 是否为外部过程
  - 若为函数，类型是什么
  - 其说明是否处理过
  - 是否递归
  - 形参是些什么

多符号表组织示例

```
class Computer {
   int cpu;
   void Crash(int numTimes) {
      int i;
      for (i = 0; i < numTimes; i = i + 1)
         Print("sad\n");
   }
}

class Mac extends Computer {
   int mouse;
   void Crash(int numTimes) {
      Print("ack!");
   }
}

class Main {
   static void main() {
      class Mac powerbook;
      powerbook = new Mac();
      powerbook.Crash(2);
   }
}
```

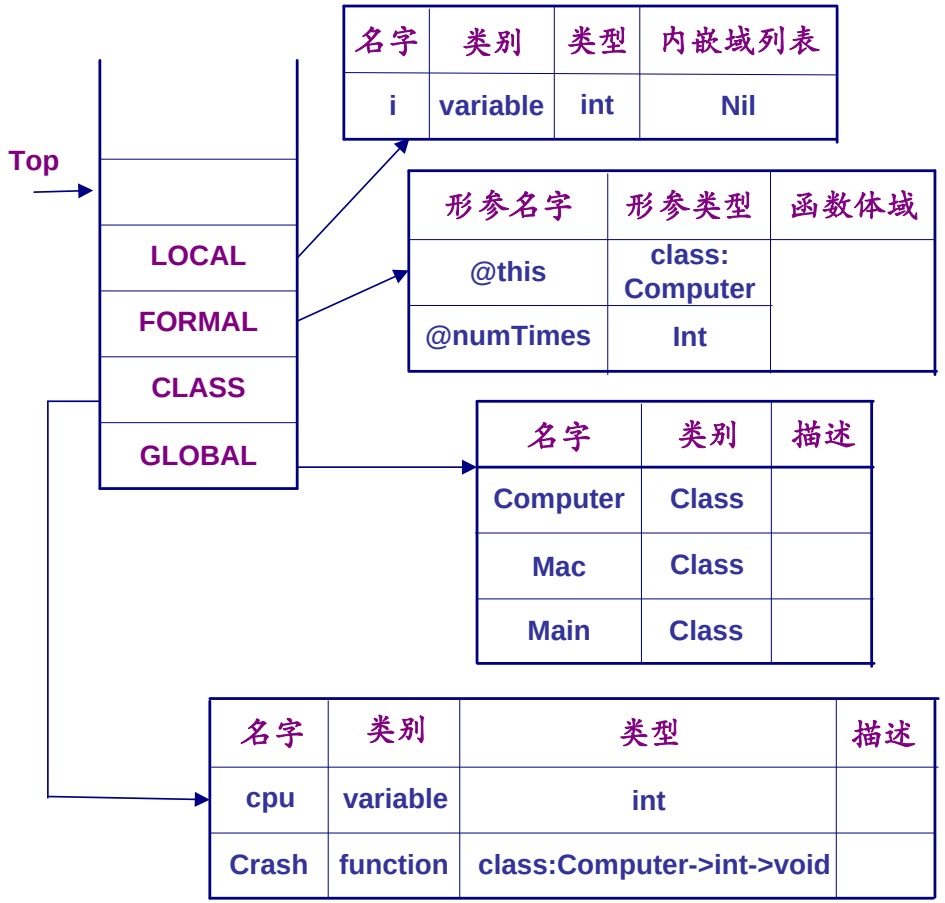


多符号表组织示例

```
class Computer {
   int cpu;
   void Crash(int numTimes) {
      int i;
      for (i = 0; i < numTimes; i = i + 1)
         Print("sad\n");
   }
}

class Mac extends Computer {
   int mouse;
   void Crash(int numTimes) {
      Print("ack!");
   }
}

class Main {
   static void main() {
      class Mac powerbook;
      powerbook = new Mac();
      powerbook.Crash(2);
   }
}
```



Summary

- 符号表的组织与作用
- 整理与查找
  - 线性表
  - 对折查找与二叉树
  - 杂凑技术（HASH技术）
- 名字的作用范围
- 符号表的内容

阅读材料：《程序设计语言编译原理（第3版）》，陈火旺等编著，国防工业出版社，2017年，第八章
《编译原理与技术》张莉等编著，第五章