

# 第四讲

## ◇ 符号表

- ◇ 符号表的作用
- ◇ 符号表的常见属性
- ◇ 符号表的实现
- ◇ 符号表体现作用域与可见性

## ◇ 符号表的作用

### — 用来存放有关标识符（符号）的属性信息

- 这些信息会在编译的不同阶段用到
- 符号表的内容将用于静态语义检查和产生中间代码
- 在目标代码生成阶段，符号表是对符号进行地址分配的依据
- 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

### — 用来体现作用域与可见性信息

## ◇ 符号的常见属性

- 符号名
- 符号的类别
- 符号的类型
- 符号的存储类别和存储分配信息
- 符号的作用域信息
- 其他属性
  - 数组内情向量
  - 记录结构的成员信息
  - 函数及过程的形参

## ◇ 符号表的实现

### — 针对符号表的常见操作

- 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

## ◇ 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

## ☆ 符号表的实现

### — 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

## ☆ 符号表体现作用域信息

- 作用域与可见性
- 作用域与符号表组织
  - 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



## ◇ 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*)

## ◇ 作用域与可见性

### – 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

## ◇ 作用域与符号表组织

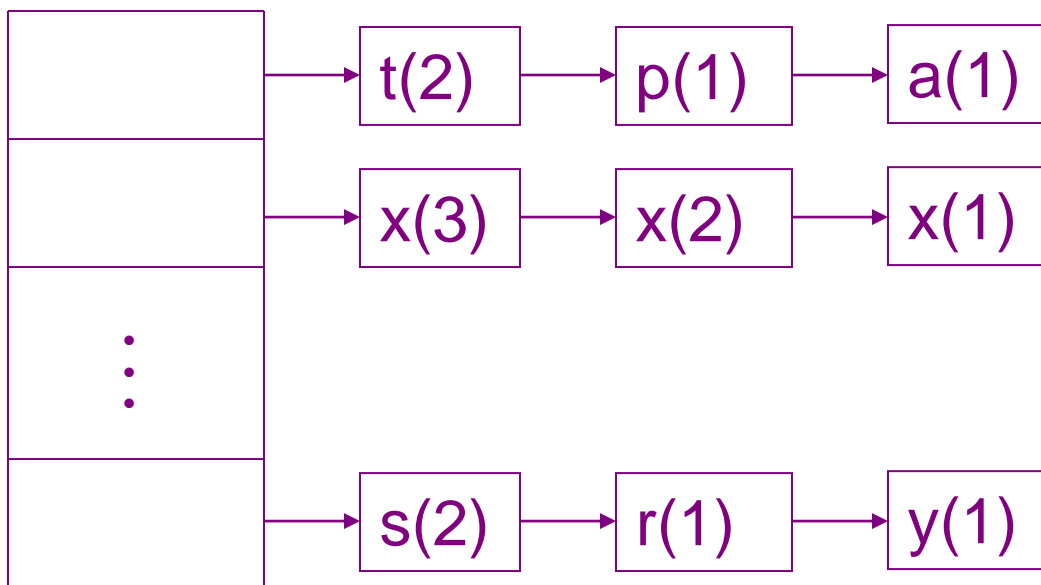
### — 作用域与单符号表组织

- 所有嵌套的作用域共用一个全局符号表
- 每个作用域有一个作用域号
- 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

# 符号表

☆ 所有嵌套的作用域共用一个全局符号表

例：右边某语言程序在处理到`/*here*/`时的符号表（以哈希表为例）



Hash Table

（表中数字代表层号）

```
const a=25;
var x,y;
procedure p;
  var z;
  begin
    .....
  end;
procedure r;
  var x, s;
  procedure t;
    var x;
    begin
      ..... /*here*/
    end;
  begin
    .....
  end;
begin
  .....
end.
```

# 符号表

☆ 所有嵌套的作用域共用一个全局符号表

例：右边某语言程序在处理到/\*here\*/时的符号表（以线性表为例）

NAME	KIND	VAL / LEVEL	ADDR	SIZE
a	CONSTANT	25		
x	VARIABLE	LEV	DX	
y	VARIABLE	LEV	DX+1	
p	PROCEDUR	LEV		CX+1
r	PROCEDUR	LEV		CX+2
x	VARIABLE	LEV+1	DX	
s	VARIABLE	LEV+1	DX+1	
t	PROCEDUR	LEV+1		CX+3

Dx: 基地址

Cx: 栈帧中控制单元数目

LEV: 层号

```
const a=25;
var x,y;
procedure p;
    var z;
    begin
        .....
    end;
procedure r;
    var x, s;
    procedure t;
        var v, x, y;
        begin
            .....
        end;
    begin /*here*/
        .....
    end;
begin
    .....
end.
```

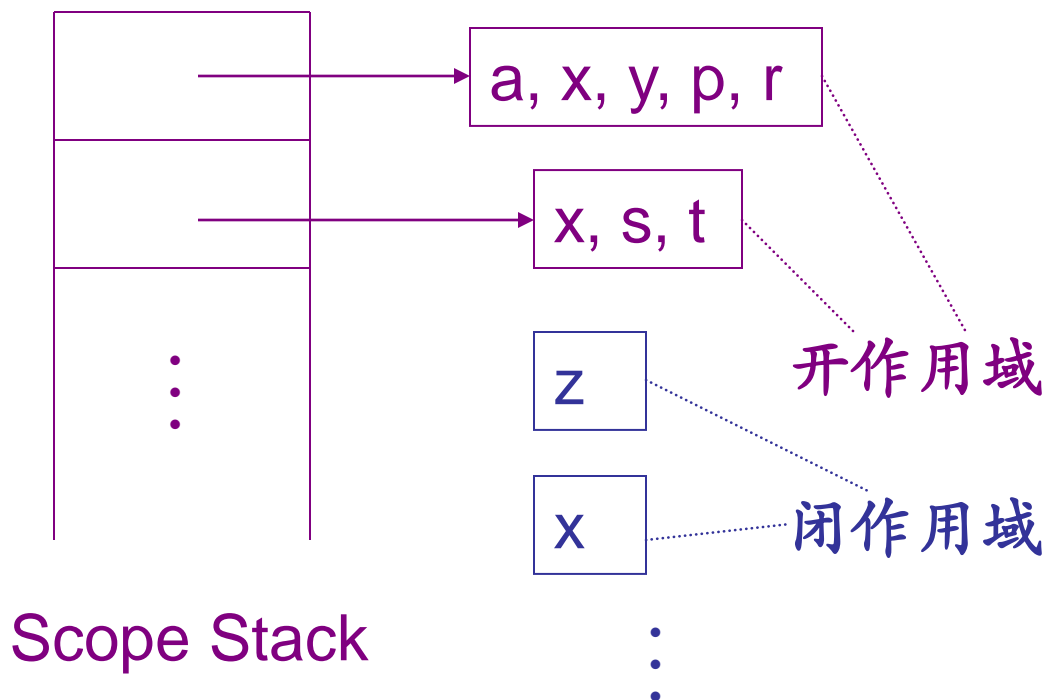
## ◇ 作用域与符号表组织

### — 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，从栈顶弹出相应的符号表

◇ 每个作用域都有各自的符号表

例：右边程序在处理到`/*here*/`时的作用域栈如下所示



```
const a=25;
var x,y;
procedure p;
  var z;
  begin
    .....
  end;
procedure r;
  var x, s;
  procedure t;
    var x;
    begin
      .....
    end;
  begin      /*here*/
    .....
  end;
begin
  .....
end.
```

# 课后作业

参见网络学堂公告：“第三次书面作业”



*That's all for today.*

*Thank You*