第四章：语义分析

符号表管理实例

1. 语义分析部分三个核心知识点

- 检查语义错误
- 抽象地址的分配（层数、偏移）
- 符号表局部化、标识符的作用域

2. 处理原则

符号表局部化处理的本质：在程序的某一个点P上，判断符号表的哪些信息是有效的

因此有以下原则：

1. 每进入一个局部化区，记录本层符号表的首地址
2. 遇到声明性标识符时，构造其语义字，查本层的符号表，检查是否有重名，有则出错，否则就把其语义字填到符号表里。
3. 遇到使用性出现，查符号表，如果查到则读取其语义字，否则出现语义错误。
4. 退出一个局部化区，'作废'本层的符号表。

3. Pascal语言的符号表的管理

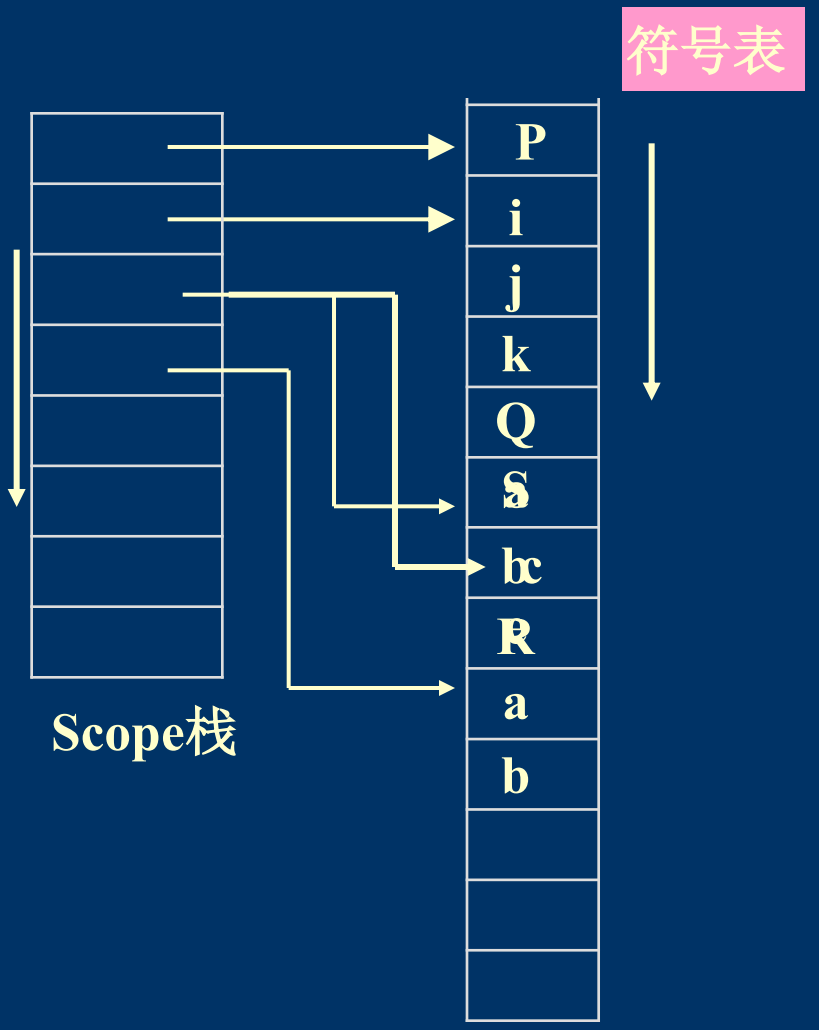
- Pascal语言的最大特点就是允许嵌套的过程声明。规定Pascal主程序的层数为0，在主程序中声明的标识符(包括过/函标识符)的层数为1，在第i层过/函中声明的标识符(包括形参和局部标识符)的层数为i+1(i≥1)。
- 用一个Scope栈来实现标识符的嵌套作用域。
- Scope栈的每项指向当前仍有效的某层符号表的首项，具体说Scope(0) 指向0层符号表的首项，Scope(i) 指向i层符号表的首项。

- "删除法"实现符号表的局部化的具体做法是：弹掉Scope栈的栈顶元素，同时释放当前层的符号表，即调整符号表区的指针。
- 假设用ℓ表示层数计数器，用s表示符号表的最后一项的地址，则删除式的局部化实现算法可描述如下：

$$s := Scope[\ell]-1$$
$$\ell := \ell - 1 ;$$

Pascal语言的符号表的实例(删除法)

```
PROGRAM example;
1.    procedure P1( )
2.          var i,j,k:integer; {t1}
3.          procedure Q2( )
4.               var a, b: real;
5.               procedure R3( )    {t2}
6.                    var a,b:boolean; {t3}
7.                    begin
8.                    R的过程体 (a,b,i)
9.                    end
10.         begin
11.              Q的过程体 (a,b,i)
12.         end;
13.         procedure S2( )
14.           var c,e:char;     {t4}
15.           begin
16.             S的过程体 (a,b,i,Q)
17.           end ;     {t5}
18.   begin
19.         P的过程体    (a,b,i,P,Q,S)
20.    end;
21  begin..P( );..end.
```



- 驻留式方法又可分为标记法和跳转法。
  - 标记法是给每个表项增加一个标记域，当某个表项变为无效时，就将该标记域设为无效；
  - 跳转法，又可分为有从前往后和从后往前两种（即上跳和下跳两种）。

- 跳转法的主要思想是（考虑从后往前的情形）：
  - 每当退出一个局部化区时，将一个跳转项添入到符号表中：

s := s +1 ; SymbTab[s]:=(# , Scope(I)-1) ;POP(Scope).

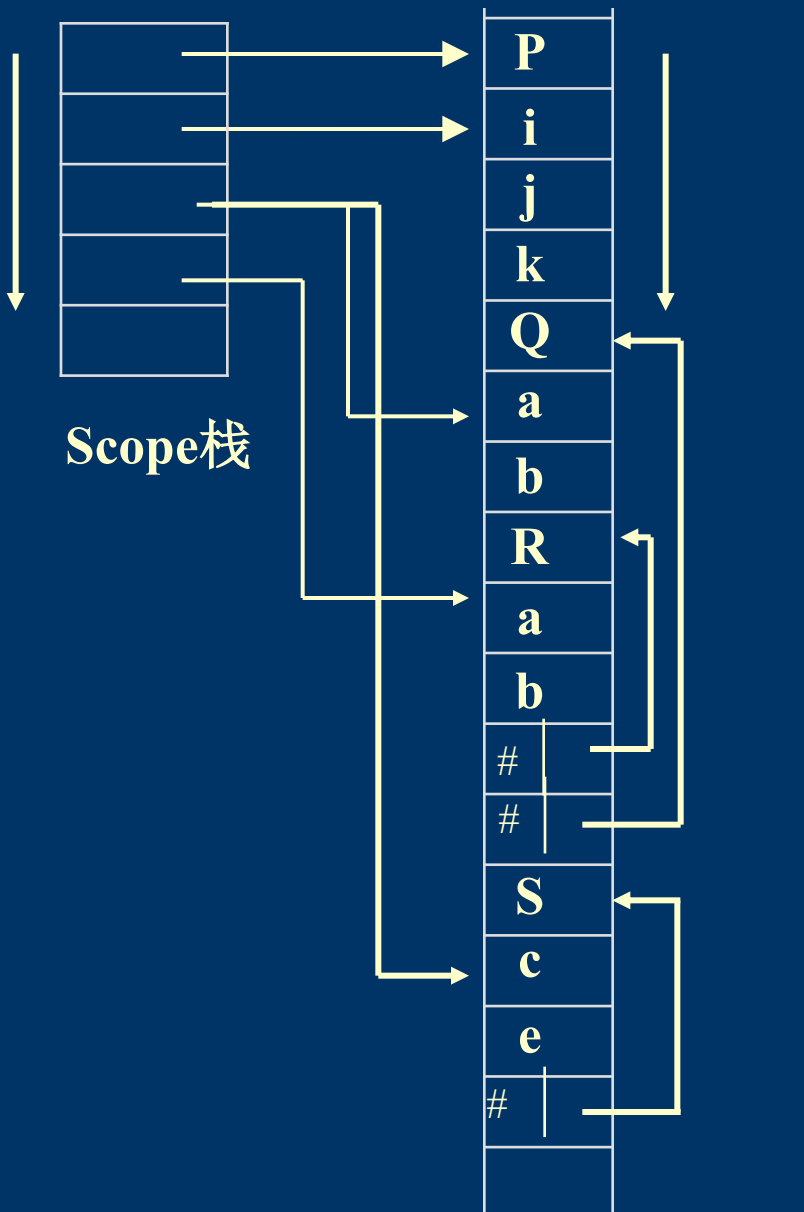
  - 当查表时，从当前符号表的末项SymbTab[s]开始往前查，当遇到跳转项时跳到跳转项的地址部分所指向的地址继续查找。

例如：如果要查的是X的属性，则查表过程大致可描述如下：(开始i 取s值)

```
[1] if  SymbTab[i].name = X then
                return(SymbTab[i].attrib);
[2] if  SymbTab[i].name = #
                then i= SymbTab[i].addr
                else i = i - 1;
[3] if  i = 0  then  Error  else  goto [1].
```

关于Pascal语言的符号表的实例（跳转法）

```
PROGRAM example;
1.    procedure P( )
2.          var i,j,k:integer; {t1}
3.          procedure Q( )
4.              var a, b: real;
5.              procedure R( )    {t2}
6.                  var a,b:boolean; {t3}
7.                  begin
8.                  R的过程体（a,b,i）
9.                  end;
10.         begin
11.             Q的过程体（a,b,i）
12.         end;
13.         procedure S( )
14.           var c,e:char;     {t4}
15.           begin
16.             S的过程体（a,b,i）
17.           end;    {t5}
18.   begin
19.           P的过程体（a,b,i）
      end
begin      .....      end.
```

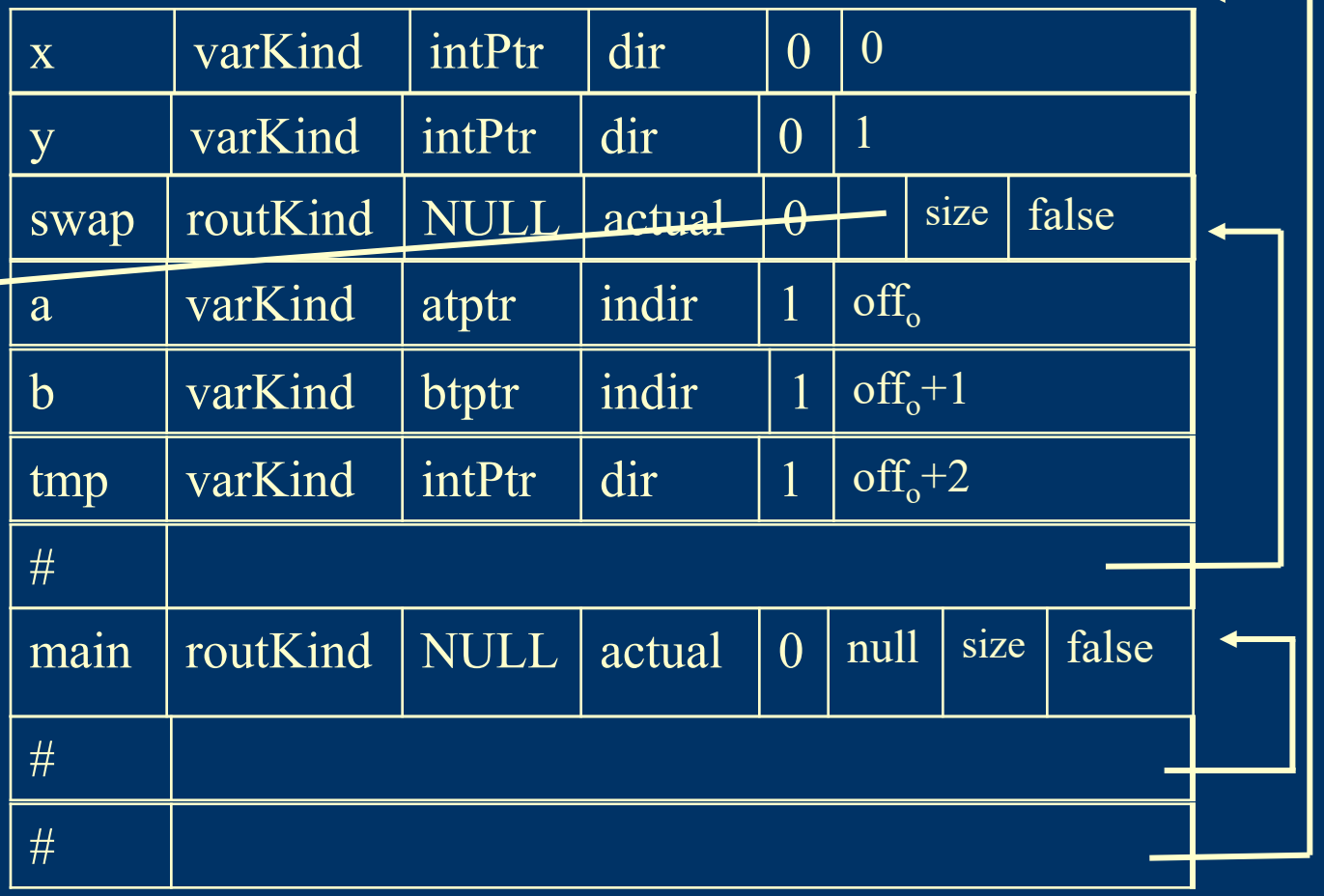


4. C语言的符号表的管理

- C语言不允许函数的嵌套声明。
- 一个C程序是由若干个函数并列组成的（其中一定要有main函数），这些函数的地位都是相同的。从main函数调用开始执行，在后面声明的函数可以调用在它之前声明的函数。
- C语言的每个标识符的层次只有两个，即0层和1层。全局标识符和所有的函数标识符的层数是0，函数的形式参数和局部标识符的层数是1。

（1） 简单的C语言符号表_驻留法

```
int x,y;
void swap(int* a, int*b )
{
   int tmp;
  tmp = *a;
  *a = *b;
  *b = tmp;
}
void main()
{
x = 10;
y = 20;
swap(&x,&y);
}
```



x	varKind	intPtr	dir	0	0		
y	varKind	intPtr	dir	0	1		
swap	routKind	NULL	actual	0		size	false
a	varKind	atptr	indir	1	off_o		
b	varKind	btptr	indir	1	off_o+1		
tmp	varKind	intPtr	dir	1	off_o+2		
#							
main	routKind	NULL	actual	0	null	size	false
#							
#							

atptr→	1	pointerTy	intPtr
btptr→	1	pointerTy	intPtr

（2） 带有分程序结构的C语言符号表

●当考虑C语言的分程序语句时，C程序符号表的构造将变得复杂一些。C语言的分程序语句(也称程序块)是由“{”和“}”括起来的程序段，其中可以包含标识符声明，还允许一个块嵌套在另一个块内。

●一个分程序就是一个局部化区。分界符"{"和"}"标记分程序的开始和结束，分界符保证分程序不是相互独立就是一个嵌在另一个里面.

带有分程序的C语言全局符号_驻留法

```
1.    main()
2.    {1
3.          int a;
4.          float b，  d;
5.      {2
6         int c;
7.         float a;
8.         {3
9.               int d;
10.              float c;
11.          {4
12.                float d;
13.             ……
14.             a=b+c+d;
15.          } 4
16.       } 3
17.       {5
18.          char d;
19.       } 5
20.     } 2
21.   } 1
```

（ 假定off（偏移）为0 ）

main	routKind	NULL	actual	0	NULL	false
a	varKind	intPtr	dir	1	0	
b	varKind	realPtr	dir	1	1	
d	varKind	realPtr	dir	1	3	
c	varKind	intPtr	dir	1	5	
a	varKind	realPtr	dir	1	6	
d	varKind	intPtr	dir	1	8	
c	varKind	realPtr	dir	1	9	
d	varKind	realPtr	dir	1	11	
#						
#						
d	varKind	charPtr	dir	1	13（8）	
#						
#						
#						

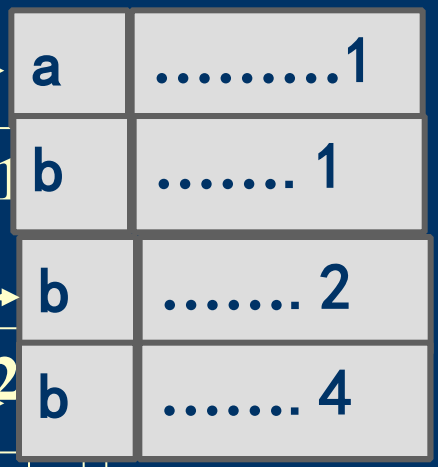
带有分程序的C语言全局符号_删除法

```
void main( )
{
    int a = 1;                          B1
    int b = 1;
    {
        int b = 2;                      B2
        {
            int a = 3;                  B3
            printf("a = %d, b = %d\n",a, b);
        }
        {
            int b = 4;                  B4
            printf("a = %d, b = %d\n",a, b);
        }
    }
    printf("a = %d, b = %d\n", a, b);
}
```

a = 3, b = 2

a = 1, b = 4

a = 1, b = 1



5. 嵌套式语言并列式语言的比较

- 特殊情形，从程序设计语言的角度来说有嵌套式语言和并列式语言
- 从处理难度角度来说，可能嵌套式语言复杂，并列式简单。 嵌套式语言里还要考虑变量运行环境。
- 并列式语言的层数分成两层，全局量定义成0层，局部量定义成1层。分符号表组织的特殊情形是分程序结构，分程序可以是嵌套的，每个分程序看成一个独立的局部化区，但这些分程序的层数是相同的。