第五章 语义分析

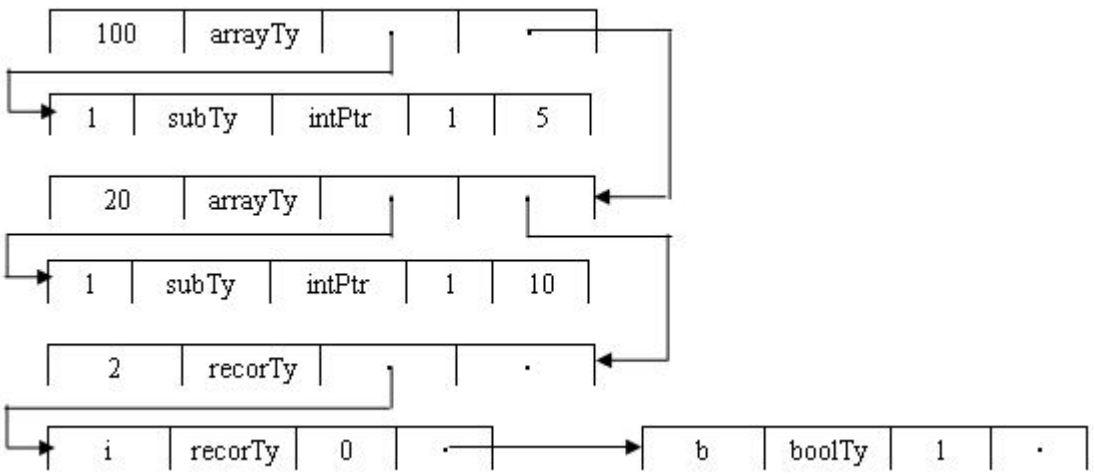
1. 试写出下面类型的内部表示：

(1) array[1..5] of array[1..10] of record i:integer;
   b:boolean
   end

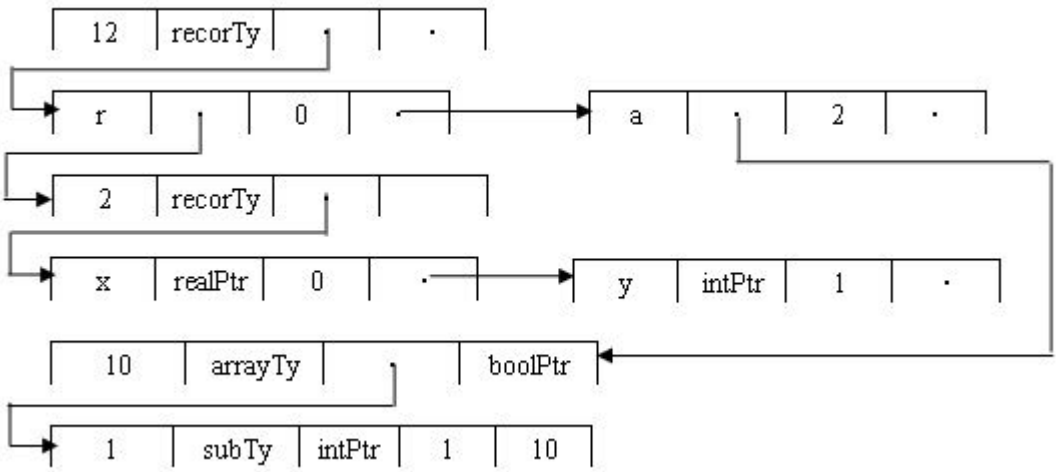
(2) record r: record x: real； y: integer end;
   a: array[1..10] of boolean
   end

(答案)

(1)



(2)



(关闭)

2． 设当前层数为 L，可用偏移量 Offset 值为 101，具有下面程序，写出本层符号表的内容。

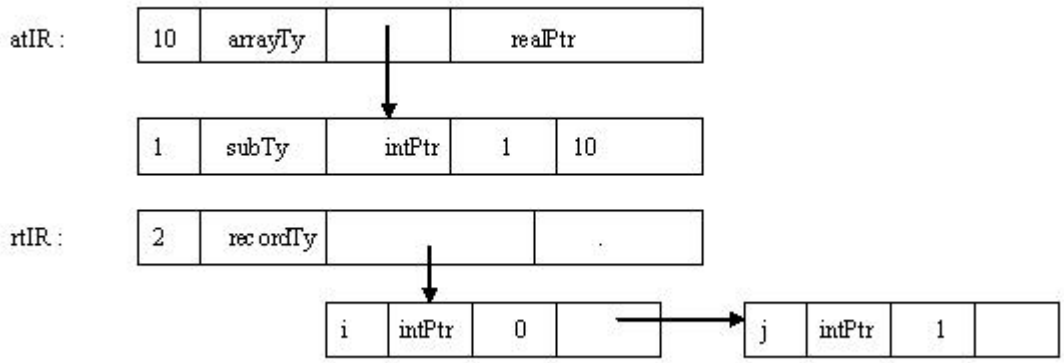
```
const m=333;
n=444;
type at=array[1..10] of real;
rt=record i,j:integer end;
VAR a,b:at;x,y:real;
```

(答案)

符号表的内容(其中 atIR 和 rtIR 表示类型的内部表示地址)：

m	intPtr	constKind	333		
n	intPtr	constKind	444		
at	atIR	typeKind	false		
rt	rtIR	typeKind	false		
a	atIR	varKind	dir	L	101
b	atIR	varKind	dir	L	111
x	realPtr	VarKind	dir	L	121
y	realPtr	varKind	dir	L	123

类型的内部表示:

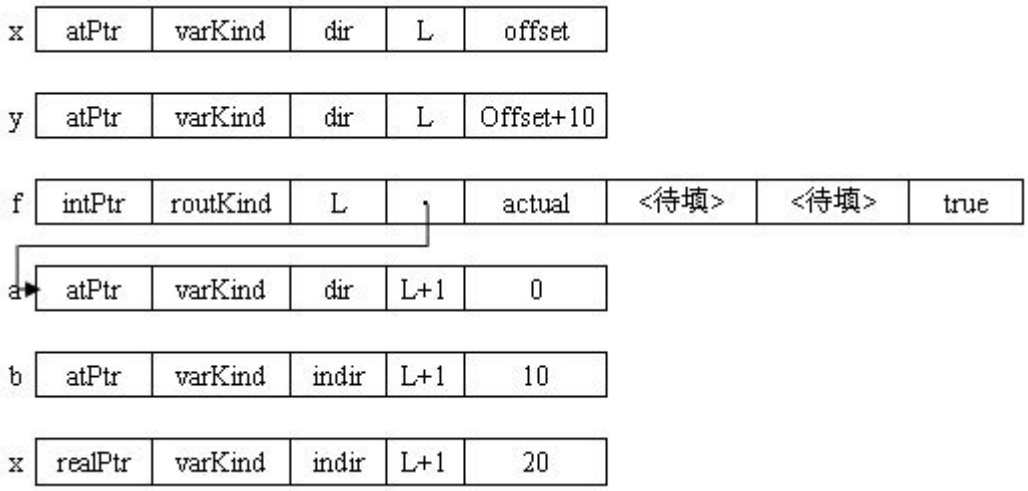


(关闭)

3. 设当前层数为 L，当前偏移量为 Offset，且有下面程序段，写出符号表内容，其中 at 见 2 题。

```
var x,y:at;
function f( a:at;
var b:at;
var x:real
):integer;
```

(答案)



(关闭)

4. 试写出标志符表的平分(折半)查表算法。

(答案)

设符号表是二叉树结构,节点描述如下:

```
SymbTable   = RECORD   id          :      String;
                Attrib       :    AttributeIR;
                Left,Right    :    ↑SymbTable
          END
```

查表算法:

```
PROCEDURE    FindEntry    (id: String;
                    VAR Entry: ↑SymbTable;
                    Var    Present : Boolean )
   {       Pesent=false;
          while( Entry!=NULL )
          {
            if (equal(id,Entry↑.id))
                    { Present=true;   return; }
```

```
            else {if (little(id,Entry↑.id)) { Entry=Entry↑.Left;}
                 else          { Entry=Entry↑.Right;}
                 }
           }

     }
```

其中,调用时 Entry 的值为二叉树的根节点指针，

equal()用来比较 id 和 Entry↑.id 是否相等(比较关键字)，若相等，返回 true，否则为 false;

little()判断 id 是否小于(比较关键字)Entry↑.id,如小于，返回 true,否则返回 false.

(关闭)

5. 试写出二叉式局部符号表的管理程序(创建，填表，查表，撤销)。

(答案)

创建：同书中 157 页 CreateTable().

撤销：同书中 157 页 DestroyTable()

填表：

```
PROCEDURE Enter (id: String, Attrib: AttribeIR;
  VAR Entry: ↑SymbTable; VAR Present: Boolean)
   { For i:= 0 TO OFF DO
   { IF (id=Table[i].id) THEN
     { present:= true;
       Entry:= Table[i] ↑;
     }
   ELSE
   present:= false;
   }
   Off:= Off + 1;
     Table[Off].id:= id; Table[Off].attrib := Attrib;
     Entry:= Table[Off] ↑;
   }
```

查表：

```
PROCEDURE Find (id: String; Flag: (One, Total);
  VAR Entry: ↑SymbTable; VAR Present: Boolean)
  { IF (Flag = One) THEN
    { For i:= 0 TO Off Do
     IF (id= Scope[Level][i].id) THEN
     { Attrib:= Scope[Level][i].attrib;
        present:= true;
```

```
    }
  ELSE {present:= false; Attrib:= NULL;}
  }
 ELSE
 { j:= CurrentL;
  While (CurrentL ≠ 0) DO
  { FOR (i:= 0 To Scale[j].Length) DO
   { IF (id = Sope[j]:[i].id) THEN
    { Attrib:= Scope[j][i].attrib;
     present:= true;
    }
   ELSE {present:= false; Attrib:= NULL;}
   }
  j:= [j/2];
  }
 }
```

(关闭)

6．试写出树式全局符号表的管理程序(创建，填表，查表，撤销)。

(答案)

树式全局符号表可采用二叉树结构，它有两种节点：

一种用于查找:

```
MidNode  = RECORD  id        :   String;
           Left,Right   :   ↑MidNode
           val      :   ↑SymbTable
         END
```

另一种保存符号表内容：

```
SymbTable  =RECORD   id        :   String;
             Attrib      :   AttributeIR;
             Next        :   ↑SymbTable
             level       :   INTEGER
           END
```

(1)创建空符号表(CreateTable),只在开始时创建一次，并且只需用 Root=new(MidNode)建立一个根节点

(2)填表(若不在表中，需要添加新节点，此时需要知道它的父节点，这里用 para 保存父节点)：

```
PROCEDURE Enter (id: String, Attrib: AttribeIR;
                 VAR Entry: ↑SymbTable; VAR Present: Boolean)
  { Var tp:↑MidNode;
    Pesent=false;
```

```
    tp=Root;para=tp
    newN=new(SymbTable);
    newN↑.id=id; newN↑.Attrib=Attrib;newN↑.level=Level;newN↑.next=NULL;
    while( tp!=NULL )
    {
        if    (equal(id,tp↑.id))
          {   Entry=tp↑.val;
            if((Entry↑.level)==Level)   { Present=true; return;}; //重复声明
              else{ InsertHead(tp,newN);  //添加表项
                  Entry=newN; return; }
          }
        else {if (little(id,tp↑.id))
                  { para=tp; tp=tp↑.Left; }  //保存父节点
            else      { para=tp; tp=tp↑.Right;}   //保存父节点
          }
     }
    tp=new(MidNode);
    tp↑.Left=tp↑.Right=NULL;tp↑.val=newN;tp↑.id=id;
    if (little(tp↑.id,tp↑.id))
                  { para↑.Left=tp;  }  //左节点
          else      { para↑.Right=tp; }  //右节点
    Entry=newN;
  }
```

(3)查表

```
PROCEDURE FindEntry (id: String; Flag: (One, Total);
                VAR Entry: ↑SymbTable; VAR Present: Boolean)
{
    Var tp:↑MidNode;
    Pesent=false;
    tp=Root  //根节点
    while( tp!=NULL )
    {
        if    (equal(id,tp↑.id))
          {   Entry=tp↑.val;
            if(((Flag==One)&&(Entry↑.level)==Level)||(flag==Total))
              { Present=true;  return; }
            else{ Present==false;return; }
          }
        else {if (little(id,tp↑.id))
                  { tp=tp↑.Left; }
            else      { tp=tp↑.Right;}
          }
```

```
        }
}
```

(4)撤销(DestroyTable)
退出一个局部化区时，需要遍历整棵树将本区的表项删除,开始时调用 DestroyTable(Root)

```
DestroyTable(Var tp:↑MidNode;)
{
    Var Entry:↑SymbTable;
    Entry=tp↑.val;
    if (Entry↑.level==Level)
       {   delete(tp,Entry)   //删除表项
         if (tp↑.val=NULL)
           { change(tp); if(p!=NULL) DestroyTable(tp);return;}
             //change()用来调整 tp 的域值，将它的 id 和相应值改为其右子树中值最小
               节点的对应值，并删除那个节点
            }
    if(tp↑.Left!=NULL) DestroyTable(tp↑.Left);
    if(tp↑.Right!=NULL) DestroyTable(tp↑.Right);
}
```

(关闭)

7. 试写出嵌套式局部符号表的管理程序(创建，填表，查表，撤销)。

(答案)

创建：同书中 157 页 CreateTable().
撤销：同书中 157 页 DestroyTable()
填表：

```
PROCEDURE Enter (id: String, Attrib: AttribeIR;
  VAR Entry: ↑SymbTable; VAR Present: Boolean)
  { For i:= 0 TO OFF DO
    { IF (id=Table[i].id) THEN
      { present:= true;
        Entry:= Table[i] ↑;
      }
      ELSE
      present:= false;
      }
    Off:= Off + 1;
    Table[Off].id:= id; Table[Off].attrib := Attrib;
```

```
  Entry:= Table[Off] ↑;
  }
查表：
PROCEDURE Find (id: String; Flag: (One, Total);
 VAR Entry: ↑SymbTable; VAR Present: Boolean)
 { IF (Flag = One) THEN
  { For i:= o TO Off Do
  IF (id= Scope[Level][i].id) THEN
  { Attrib:= Scope[Level][i].attrib;
    present:= true;
    }
  ELSE {present:= false; Attrib:= NULL;}
  }
 ELSE
 { j:= LEVEL;
  While (j≠ 0) DO
  { FOR (i:= 0 TO Scale[j].length) DO
    IF (id = Scope[j][i].id) THEN
      { Attrib:= Scope[j][i].attrib;
        present:= true;
      }
    ELSE {present:= false; Attrib:= NULL;}
  }
 j:= j-1;
 }
}
```

(关闭)

8. 在我们这里标号分为声明性、定位性和使用性符号，试回答下列问题：

(1) 如何检查无定位性标号的错误？
(2) 如何检查重复定位的错误？
(3) 如何检查 goto 语句跳入结构语句内部的错误？
(4) 如何检查 goto 语句跳入过程内部的错误？

(答案)

标号的处理见书 P171 的标号语义分析原理：
其中用到三个表：标号声明表(LDEC)，标号定位表(LDEF)，和标号使用表(LUSE)
考虑这三个表的情况，就可以检查这四种错误：
(1)

(2)
(3)
(4)

(关闭)

9. 类型等价有按名等价和按结构的等价，试同其实现有什么主要区别？

(答案)

按名字等价	实现方法
1．类型为一个类型名	判断其名字是否相同
2．类型为结构类型	对结构结构类型的内部表示进行判断是否按名字等价
按结构等价	**实现方法**
	对结构结构类型的内部表示进行判断是否按结构等价

(关闭)

10．每个类型都有其空间长度，我们为了简单起见，主要考虑了以一个存储单元为存储单位。但实际上，有些类
以一个 bit 或一个 byte 为单位，有些类型可能以一个短单元或一个长单元为存储单位，这时应怎么处理？

(答案)

(无)

(关闭)