

# 第1讲习题

## 习题1.1

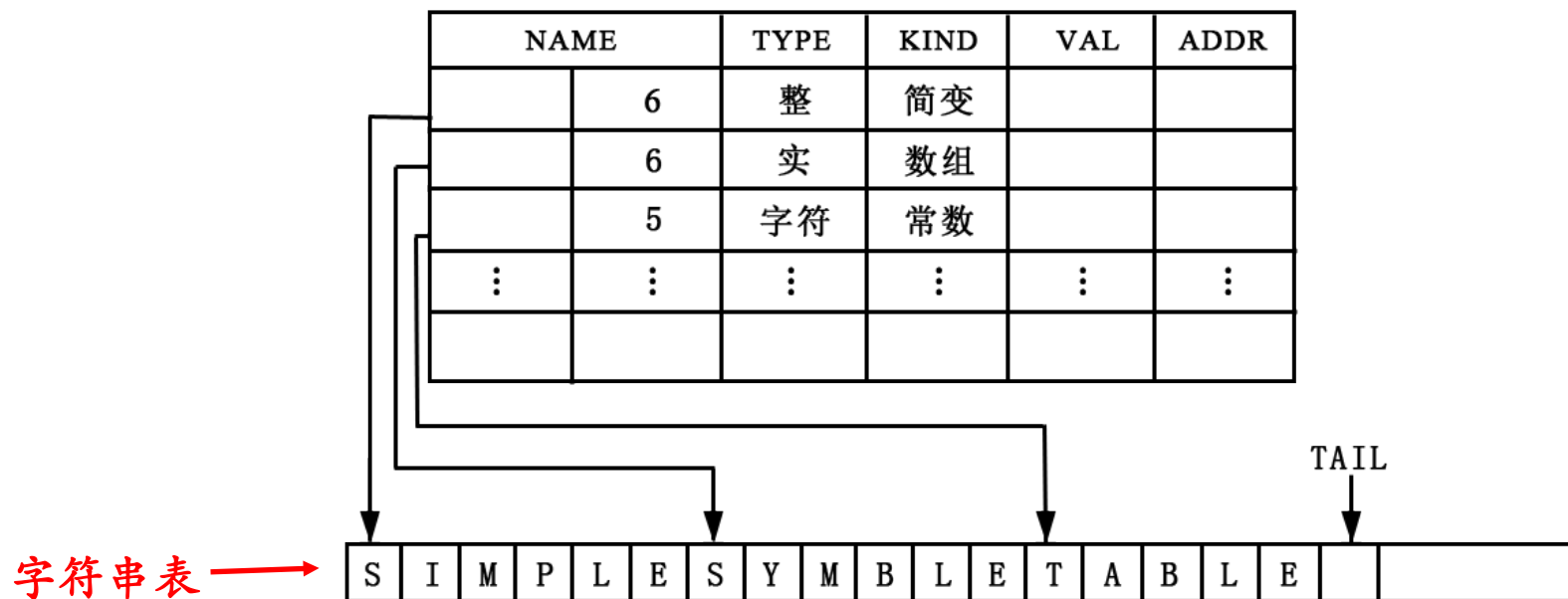
- 将下面的C++程序划分成正确的词素 (token) 序列。哪些词素应该有相关联的属性值？应该具有什么值？

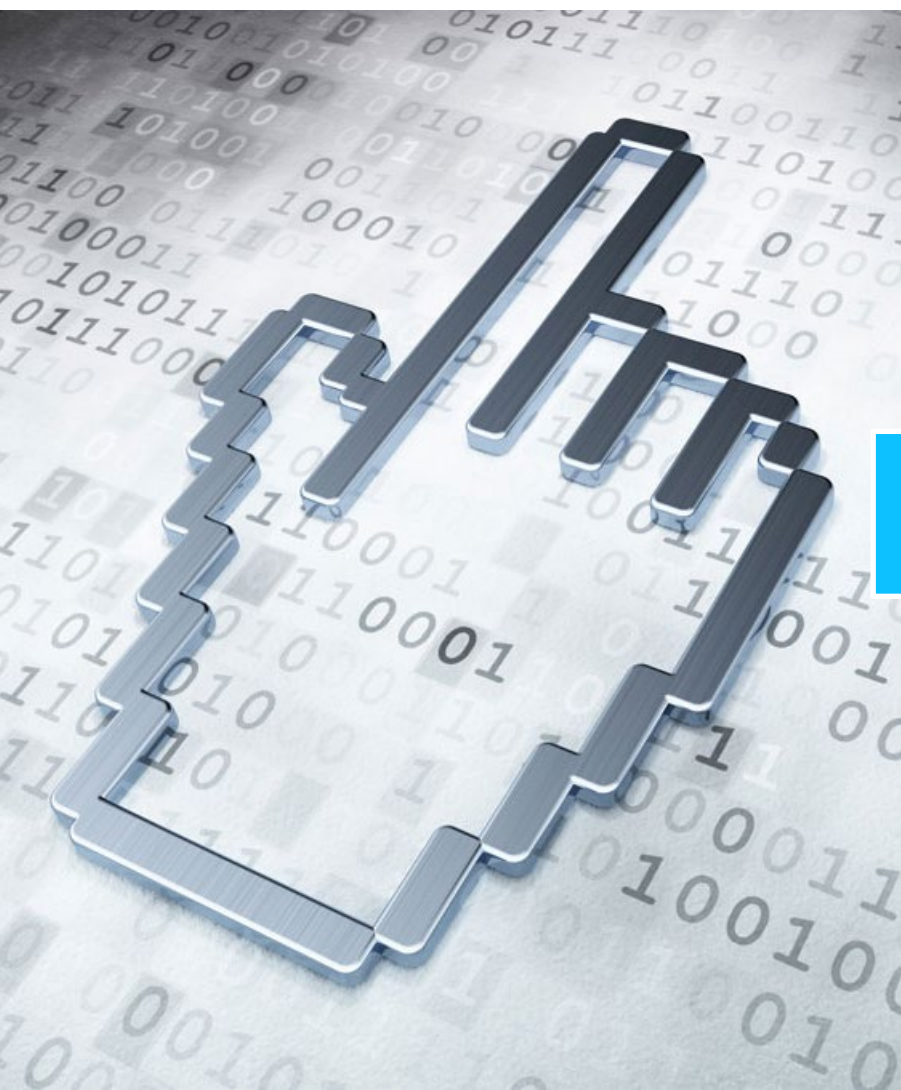
```
float limitedSquare(x) {float x ;  
/* returns x-squared, but never more than 100 */  
return (x<=-10.01 || x>=10.0)?100:x*x;  
}
```

## 习题1.2

- 符号表中NAME字段为什么要设计字符串表这样一种数据结构？而不是把标识符对应的字符串直接存放到NAME字段

符号表(Symbol Table)





## 第2讲习题

## 习题2.1

➤ 考虑上下文无关文法：

$$S \rightarrow S S + \mid S S * \mid a$$

以及串  $aa+a^*$ 。

- ① 给出这个串的一个最左推导。
- ② 给出这个串的一个最右推导。
- ③ 给出这个串的一棵语法分析树。
- ④ 该文法生成的语言是什么？
- ⑤ 这个文法是否是二义性的？

## 习题2.2

### ➤对下列各文法重复习题2.1

➤ (1)  $S \rightarrow 0 S 1 \mid 0 1$

和串 000111

➤ (2)  $S \rightarrow + S S \mid * S S \mid a$

和串  $+ * a a a$

➤ (3)  $S \rightarrow S (S) S \mid \varepsilon$

和串  $((())())$

➤ (4)  $S \rightarrow a \mid S + S \mid S S \mid S * \mid (S)$

和串  $(a + a) * a$

➤ (5)  $S \rightarrow (L) \mid a$  以及  $L \rightarrow L, S \mid S$  和串  $((a, a), a, (a))$

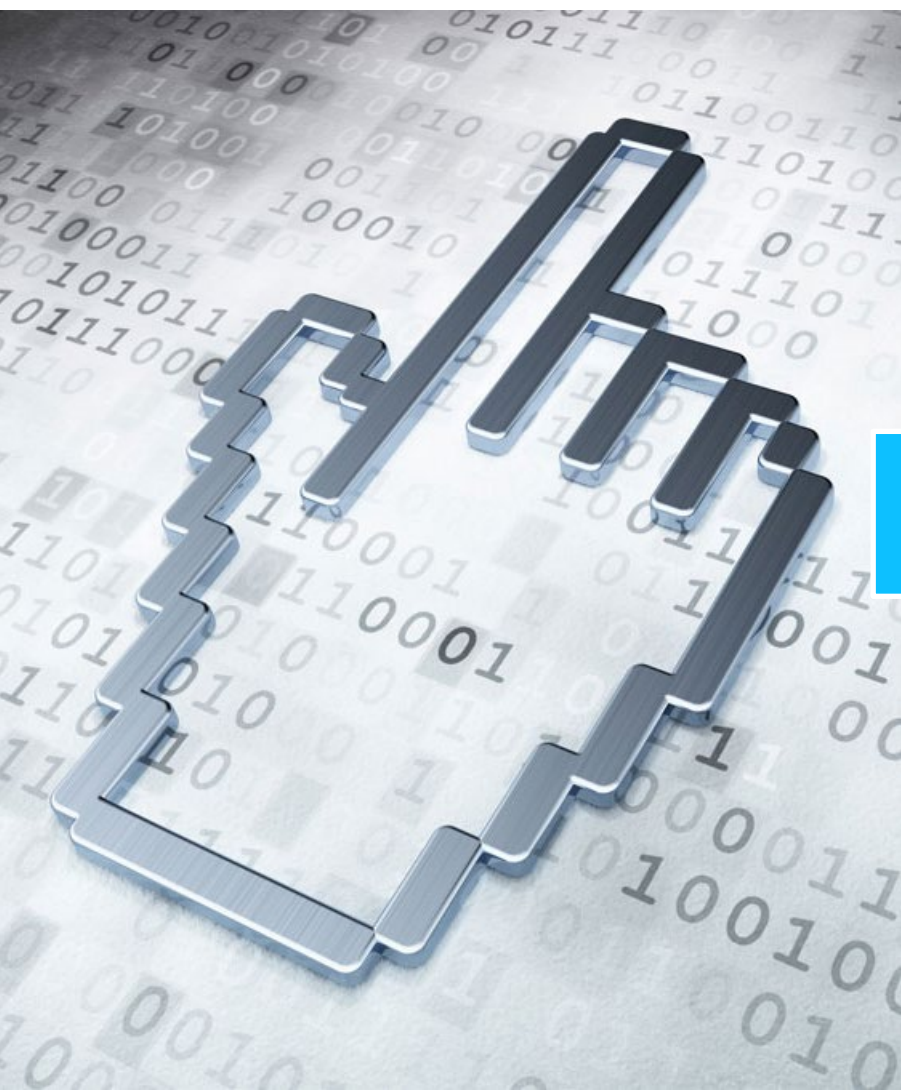
➤ (6)  $S \rightarrow a S b S \mid b S a S \mid \varepsilon$

和串 aabbab

➤ (7)  $E \rightarrow E \text{ or } T \mid T$

$T \rightarrow T \text{ and } F \mid F$

$F \rightarrow \text{not } F \mid (E) \mid \text{true} \mid \text{false}$  (只需完成第④⑤两题)

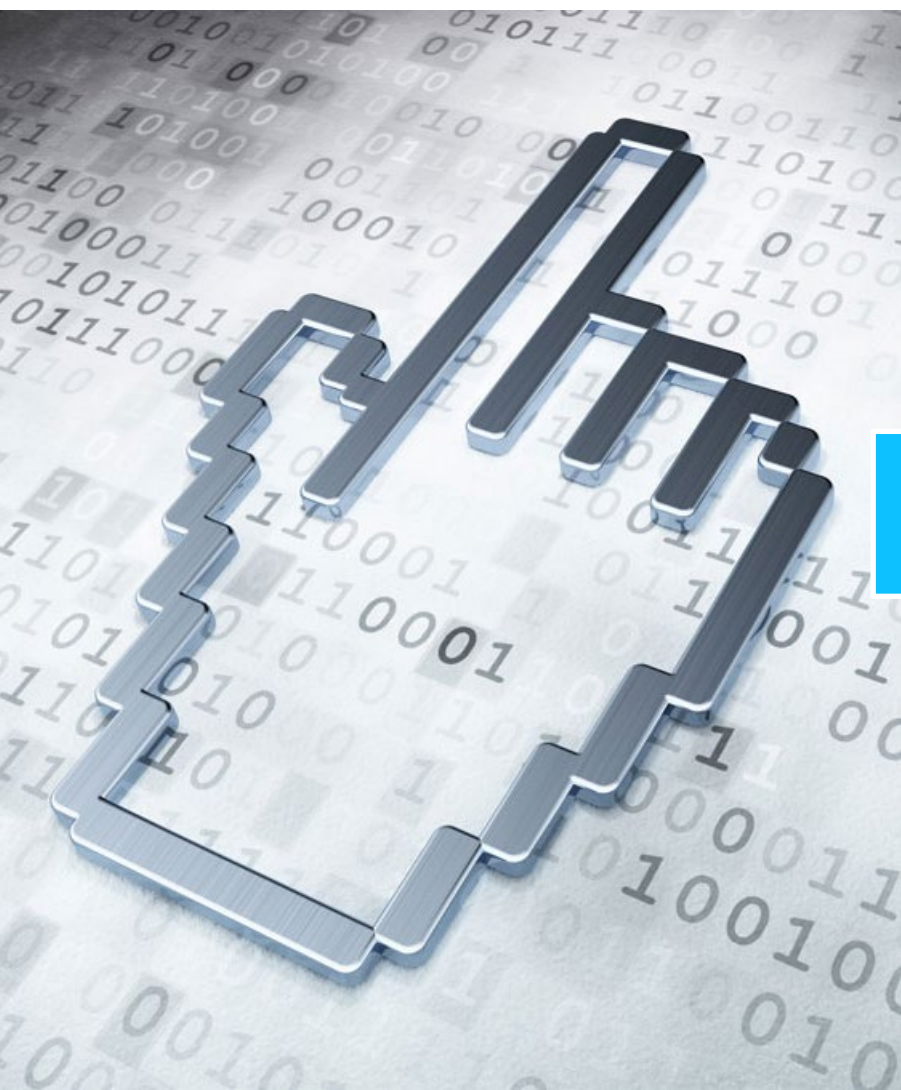


## 第3讲习题

## 习题3.1

- 写出“注释”的正则定义，并为其设计一个DFA。  
注释是/\*和\*/之间的串，且串中没有不在双引号（"）中的\*/





## 第4讲习题

## 习题4.1

➤ 为下面的每一个文法设计一个预测分析器，并给出预测分析表。你可能先要对文法进行提取左公因子或消除左递归的操作。计算各文法的FOIRST和FOLLOW集合。

➤ (1)  $S \rightarrow 0 S 1 \mid 0 1$

➤ (2)  $S \rightarrow + S S \mid * S S \mid a$

➤ (3)  $S \rightarrow S (S) S \mid \varepsilon$

➤ (4)  $S \rightarrow a \mid S + S \mid S S \mid S * \mid (S)$

➤ (5)  $S \rightarrow ( L ) \mid a$  以及  $L \rightarrow L , S \mid S$

➤ (6)  $E \rightarrow E \text{ or } T \mid T$

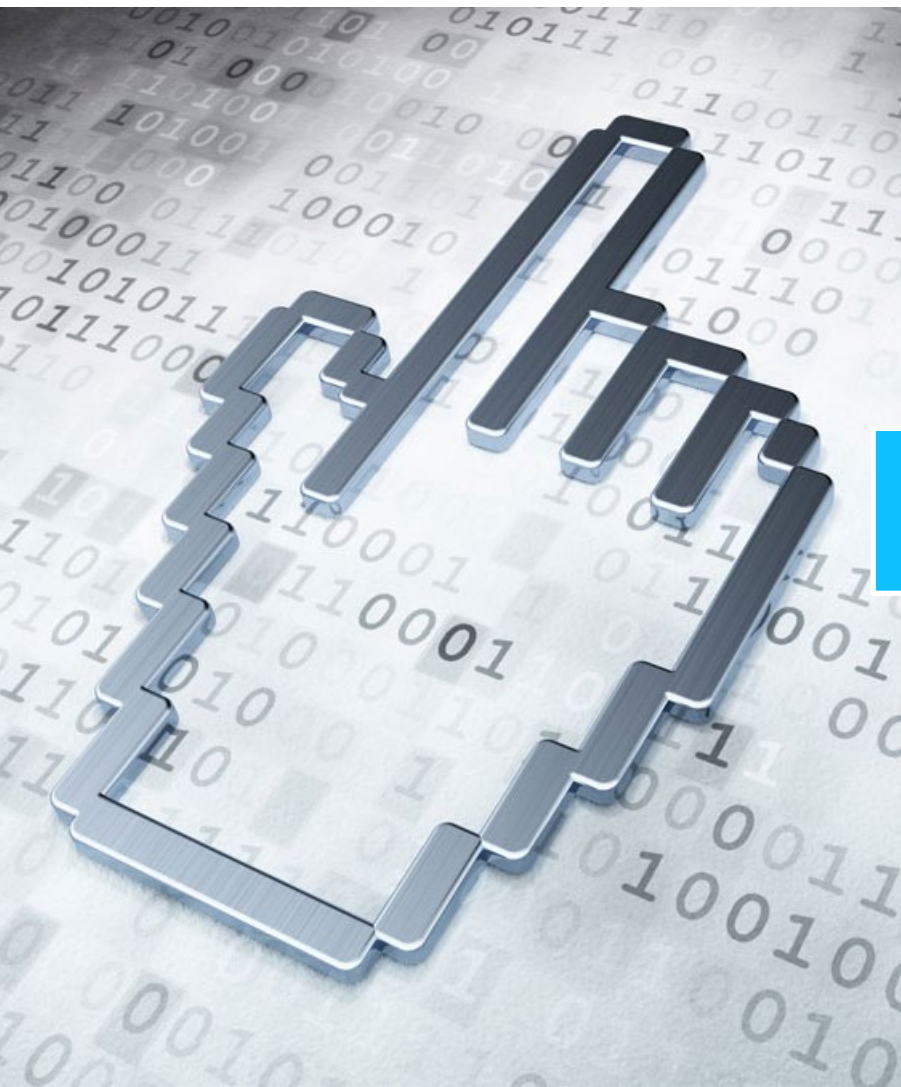
$T \rightarrow T \text{ and } F \mid F$

$F \rightarrow \text{not } F \mid ( E ) \mid \text{true} \mid \text{false}$

## 习题4.2

- 为下面的文法设计一个预测分析器，并给出预测分析表。你可能先要对文法进行提取左公因子或消除左递归的操作。计算文法的FIRST和FOLLOW集合。

$$S \rightarrow S S + \mid S S * \mid a$$



## 第5讲习题

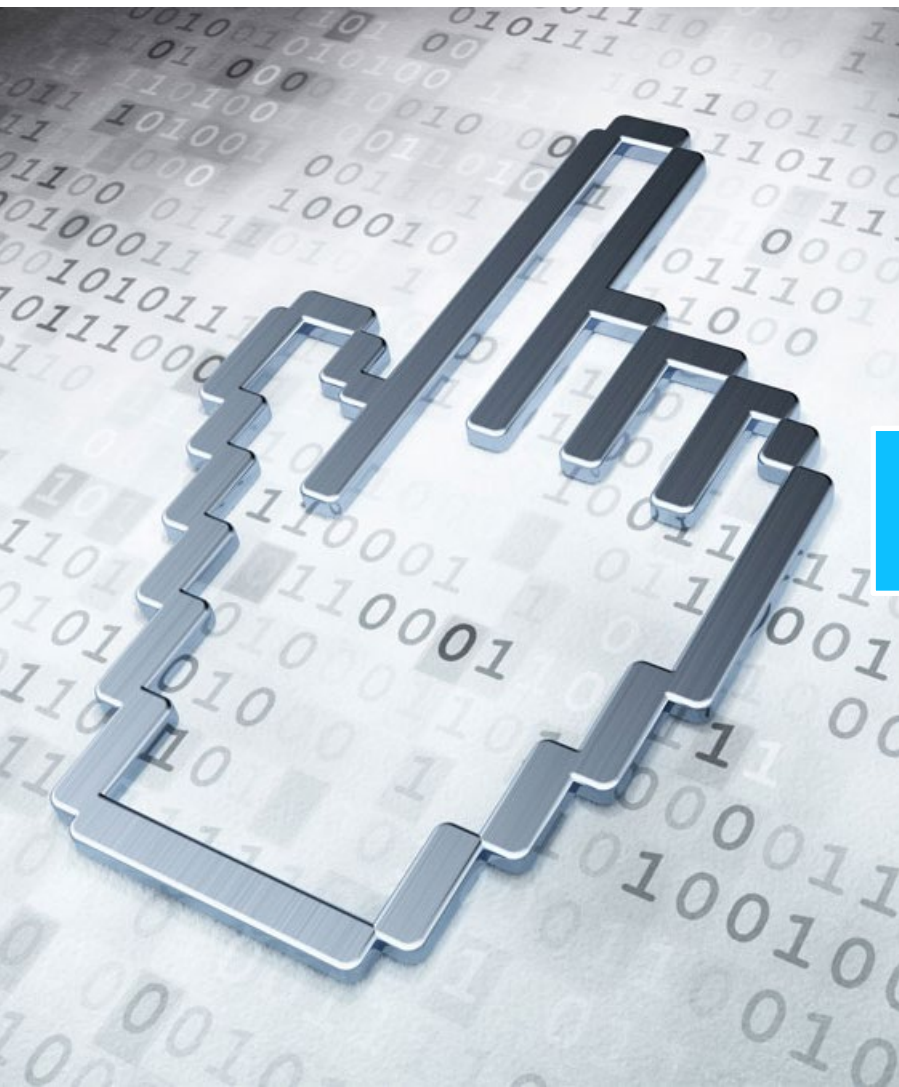
## 习题5.1

➤ 为下列文法构造递归下降语法分析器（参见SPOC讲义“第4章 语法分析 - 上.pdf”第42~48页）

➤ (1)  $S \rightarrow + S S \mid - S S \mid a$

➤ (2)  $S \rightarrow S ( S ) S \mid \varepsilon$

➤ (3)  $S \rightarrow 0 S 1 \mid 0 1$



## 第6讲习题

## 习题6.1

➤ 对于文法  $S \rightarrow 0S1 \mid 01$ ，指出下面各个最右句型的句柄：

➤ (1) 000111

➤ (2) 00S11

## 习题6.2

➤ 对于文法  $S \rightarrow SS+ | SS* | a$ ，指出下面各个最右句型的句柄：

➤ (1)  $SSS+a*+$

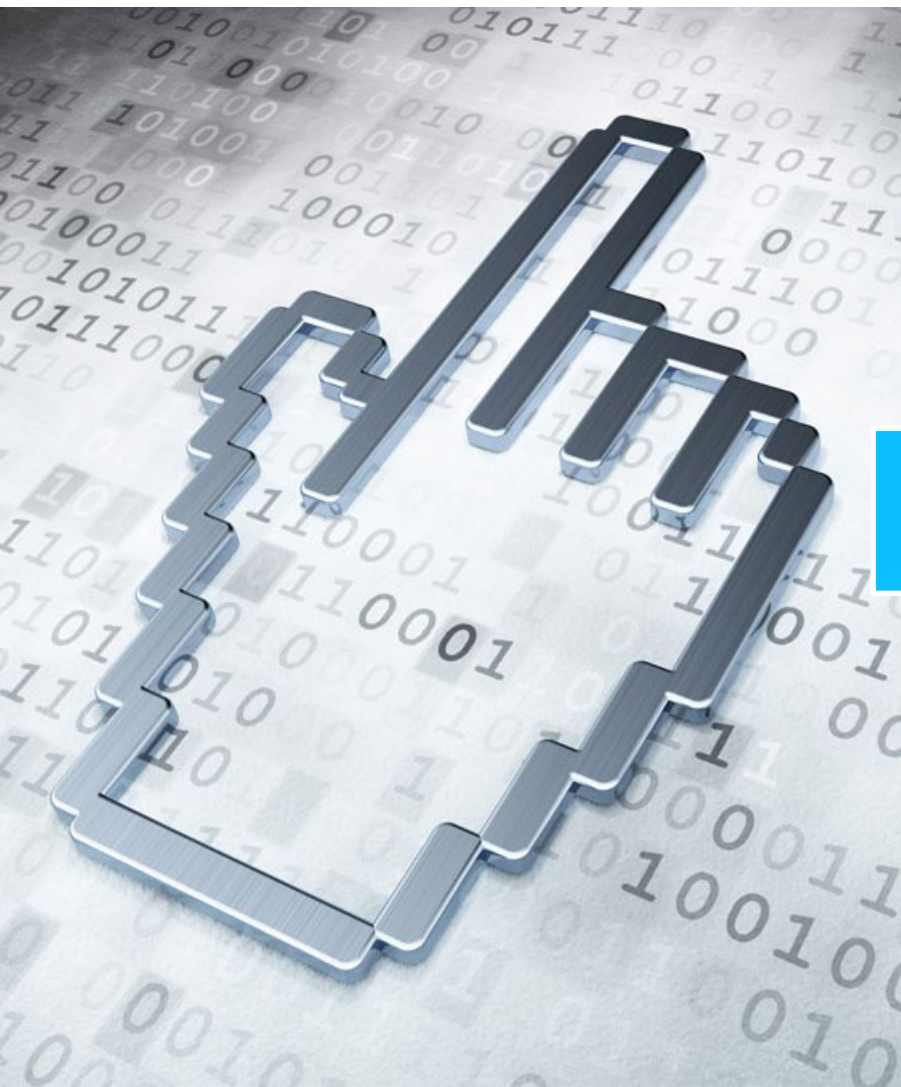
➤ (2)  $SS+a*a+$

➤ (3)  $aaa*a++$



## 习题6.3

- 我们可以根据语法分析栈中的LR状态来推断出这个状态表示了什么文法符号。我们如何推导出这个信息？



## 第7讲习题

## 习题7.1

➤ 为（增广）文法

$$S \rightarrow S S + \mid S S * \mid a$$

构造SLR项集。计算这些项集的GOTO函数。给出这个文法的语法分析表。这个文法是SLR文法吗？如果是，利用得到的分析表，给出处理输入 $aa*a+$ 时的各个动作。

## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (1)  $S \rightarrow 0 S 1 \mid 0 1$                   输入样例：000111

## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (2)  $S \rightarrow + S S \mid * S S \mid a$     输入样例：+ \* a a a

## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (3)  $S \rightarrow S(S)S \mid \varepsilon$                   输入样例：((()())

## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (4)  $S \rightarrow a \mid S + S \mid S S \mid S * \mid (S)$     输入样例： $(a + a) * a$

## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (5)  $S \rightarrow (L) \mid a$  以及  $L \rightarrow L, S \mid S$

输入样例：((a, a), a, (a))



## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (6)  $S \rightarrow a S b S \mid b S a S \mid \varepsilon$       输入样例：aabbab

## 习题7.2

➤ 对于下列各（增广）文法：

① 构造SLR项集和它们的GOTO函数。

② 指出你的项集中有没有动作冲突。

③ 如果存在SLR语法分析表，构造出这个语法分析表。

➤ (7)  $E \rightarrow E \text{ or } T \mid T$

$T \rightarrow T \text{ and } F \mid F$

$F \rightarrow \text{not } F \mid ( E ) \mid \text{true} \mid \text{false}$

## 习题7.3

➤说明下面的文法

$$S \rightarrow A a A b \mid B b B a$$

$$A \rightarrow \varepsilon$$

$$B \rightarrow \varepsilon$$

是LL(1)的，但不是SLR(1)的

## 习题7.4

➤说明下面的文法

$$S \rightarrow S A \mid A$$

$$A \rightarrow a$$

是SLR(1)的，但不是LL(1)的

## 习题7.5

➤ 下面是一个二义性文法：

$$S \rightarrow A S \mid b$$

$$A \rightarrow S A \mid a$$

构造出这个文法的规范LR项集族。如果我们试图为这个文法构造出一个LR语法分析表，必然会存在某些冲突动作。都有哪些冲突动作？假设我们使用这个语法分析表，并且在出现冲突时不确定地选择一个可能的动作。给出处理输入abab时的做有可能的动作序列。

## 习题7.6

➤ 为文法  $S \rightarrow S S + \mid S S * \mid a$  构造

① 规范LR项集族。

② LALR项集族。

## 习题7.7

➤ 对于下列各（增广）文法，构造

① 规范LR项集族

② LALR项集族

➤ (1)  $S \rightarrow 0 S 1 \mid 0 1$

➤ (2)  $S \rightarrow + S S \mid * S S \mid a$

➤ (3)  $S \rightarrow S (S) S \mid \varepsilon$

➤ (4)  $S \rightarrow ( L ) \mid a$  以及  $L \rightarrow L , S \mid S$

## 习题7.8

➤说明下面的文法

$$S \rightarrow A a \mid b A c \mid d c \mid b d a$$

$$A \rightarrow d$$

是LALR(1)的，但不是SLR(1)的。



## 习题7.9

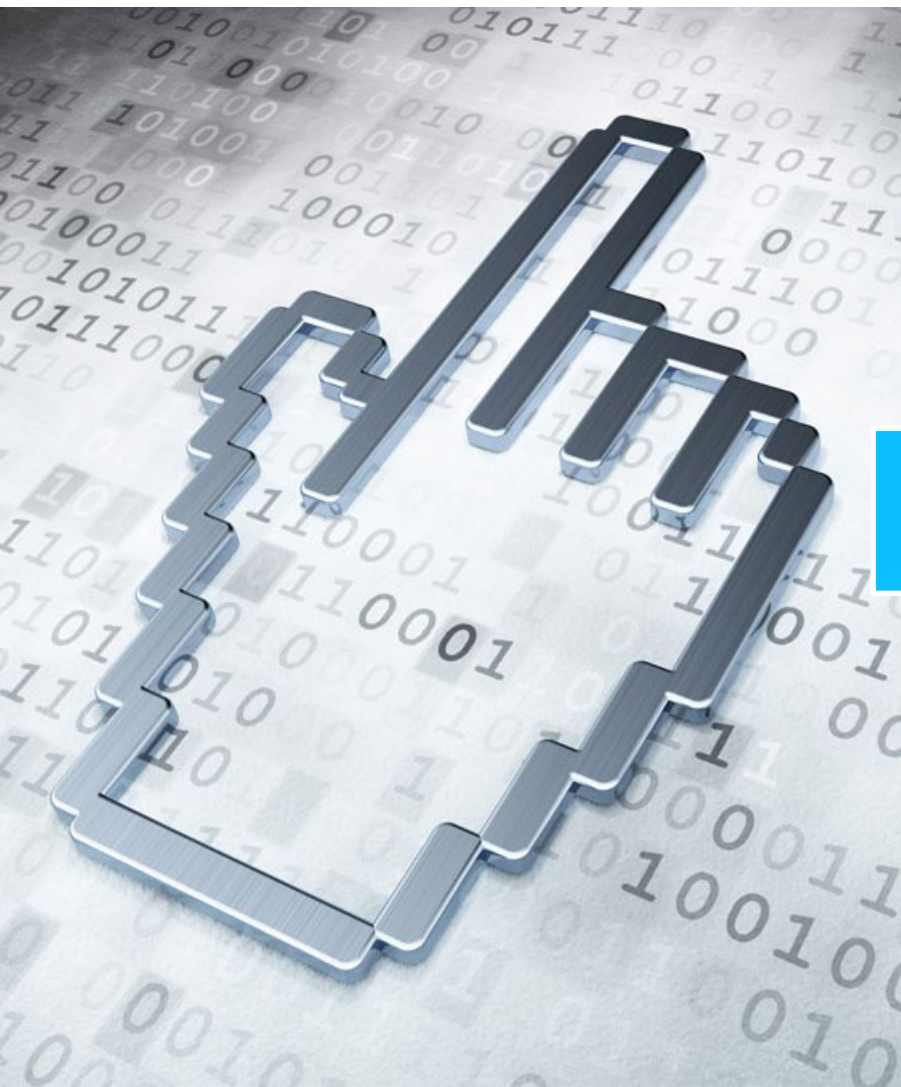
➤说明下面的文法

$$S \rightarrow A a \mid b A c \mid B c \mid b B a$$

$$A \rightarrow d$$

$$B \rightarrow d$$

是LR(1)的，但不是LALR(1)的。



## 第8讲习题

## 习题8.1

➤ 对于下图所示的SDD，给出下列表达式对应的注释语法分析树：

产生式	语义规则
1) $L \rightarrow E \text{ n}$	$L.val = E.val$
2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3) $E \rightarrow T$	$E.val = T.val$
4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
5) $T \rightarrow F$	$T.val = F.val$
6) $F \rightarrow ( E )$	$F.val = E.val$
7) $F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

➤ (1)  $(3+4)*(5+6)\text{n}$

➤ (2)  $1*2*3*(4+5)\text{n}$

➤ (3)  $(9+8*(7+6)+5)*4\text{n}$

## 习题8.2

➤ 给定图1所示的SDD，图2是句子 $3*5$ 的注释分析树的依赖图。图2的全部拓扑顺序有哪些？

产生式	语义规则
1) $T \rightarrow F T'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \rightarrow * F T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3) $T' \rightarrow \epsilon$	$T'.syn = T'.inh$
4) $F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

图1

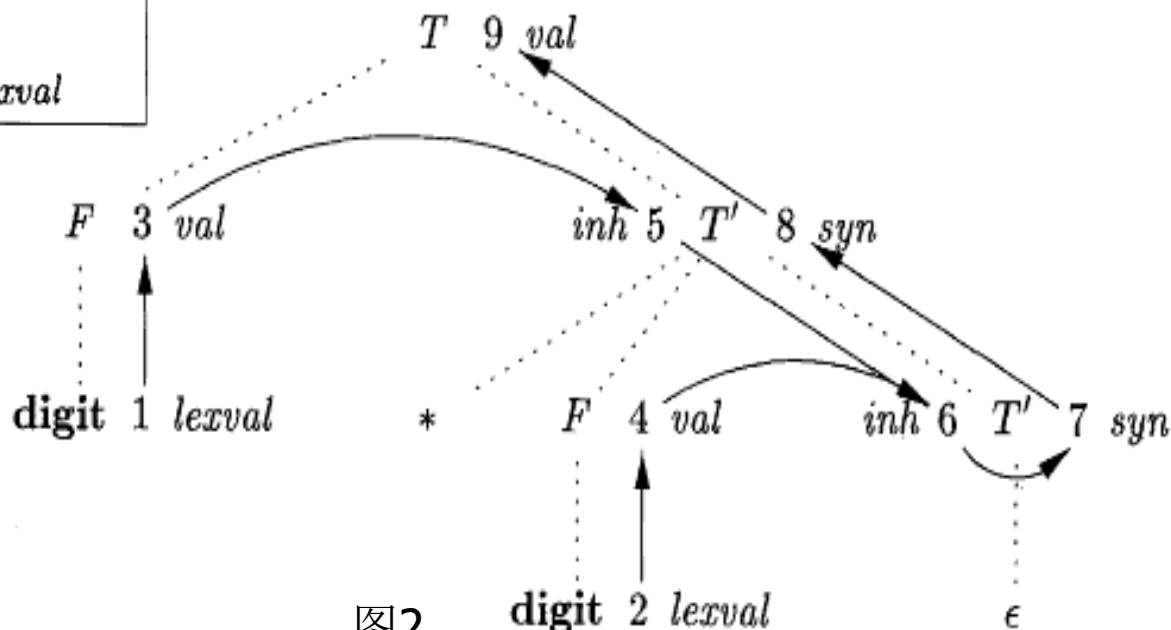


图2

## 习题8.3

➤ 对于下图中的SDD，给出下列表达式对应的注释语法分析树：

➤ (1) `int a,b,c`

➤ (2) `real w,x,y,z`

	产生式	语义规则
(1)	$D \rightarrow T L$	$L.inh = T.type$
(2)	$T \rightarrow \text{int}$	$T.type = \text{int}$
(3)	$T \rightarrow \text{real}$	$T.type = \text{real}$
(4)	$L \rightarrow L_1, \text{id}$	$L_1.inh = L.inh$ $addtype(id.lexeme, L.inh)$
(5)	$L \rightarrow \text{id}$	$addtype(id.lexeme, L.inh)$

## 习题8.4

➤假设我们有一个产生式 $A \rightarrow BCD$ 。A、B、C、D这四个非终结符都有两个属性：s是一个综合属性，而i是一个继承属性。对于下面的每组规则，指出

- ① 这些规则是否满足S属性定义的要求。
- ② 这些规则是否满足L属性定义的要求。
- ③ 是否存在和这些规则一致的求值过程？

➤(1)  $A.s = B.i + C.s$

## 习题8.4

➤假设我们有一个产生式 $A \rightarrow BCD$ 。A、B、C、D这四个非终结符都有两个属性：s是一个综合属性，而i是一个继承属性。对于下面的每组规则，指出

- ① 这些规则是否满足S属性定义的要求。
- ② 这些规则是否满足L属性定义的要求。
- ③ 是否存在和这些规则一致的求值过程？

➤(2)  $A.s = B.i + C.s$

$D.i = A.i + B.s$

## 习题8.4

➤假设我们有一个产生式 $A \rightarrow BCD$ 。A、B、C、D这四个非终结符都有两个属性：s是一个综合属性，而i是一个继承属性。对于下面的每组规则，指出

- ① 这些规则是否满足S属性定义的要求。
- ② 这些规则是否满足L属性定义的要求。
- ③ 是否存在和这些规则一致的求值过程？

➤(3)  $A.s = B.s + D.s$



## 习题8.4

➤假设我们有一个产生式 $A \rightarrow BCD$ 。A、B、C、D这四个非终结符都有两个属性：s是一个综合属性，而i是一个继承属性。对于下面的每组规则，指出

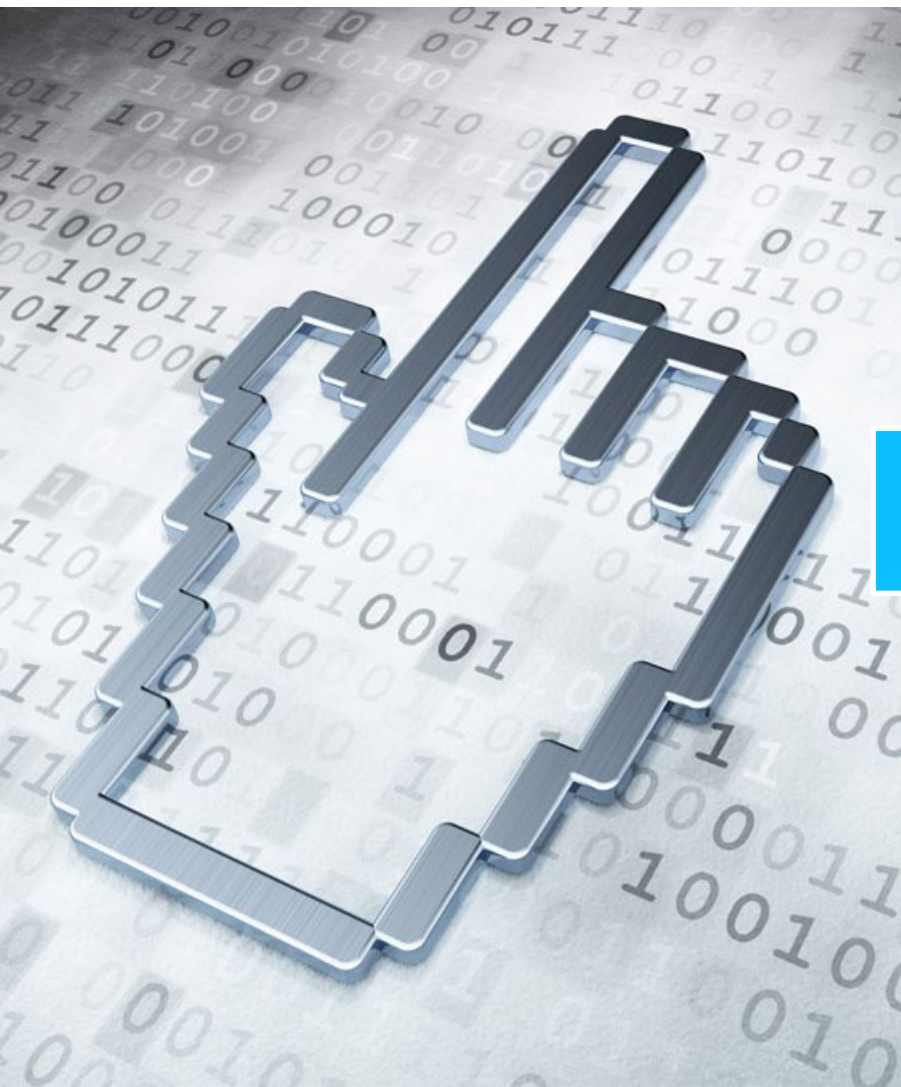
- ① 这些规则是否满足S属性定义的要求。
- ② 这些规则是否满足L属性定义的要求。
- ③ 是否存在和这些规则一致的求值过程？

➤(4)  $A.s = D.i$

$$B.i = A.s + C.s$$

$$C.i = B.s$$

$$D.i = B.i + C.i$$



## 第9讲习题

## 习题9.1

- 下图中的SDD计算诸如 $3*5$ 和 $3*5*7$ 这样的项。扩展下图中的SDD，使它可以像习题8.1图中所示的那样处理表达式。

产生式	语义规则
1) $T \rightarrow F T'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \rightarrow * F T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3) $T' \rightarrow \epsilon$	$T'.syn = T'.inh$
4) $F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

## 习题9.2

➤这个文法生成了含“小数点”的二进制数：

$$S \rightarrow L . L \mid L$$

$$L \rightarrow L B \mid B$$

$$B \rightarrow 0 \mid 1$$

设计一个**L属性**的SDD来计算S.val，即输入串的十进制数值。比如，串101.101应该被翻译成十进制数5.625。提示：使用一个继承属性L.side来指明一个二进制位在小数点的那一边。

## 习题9.3

➤这个文法生成了含“小数点”的二进制数：

$$S \rightarrow L.L \mid L$$

$$L \rightarrow LB \mid B$$

$$B \rightarrow 0 \mid 1$$

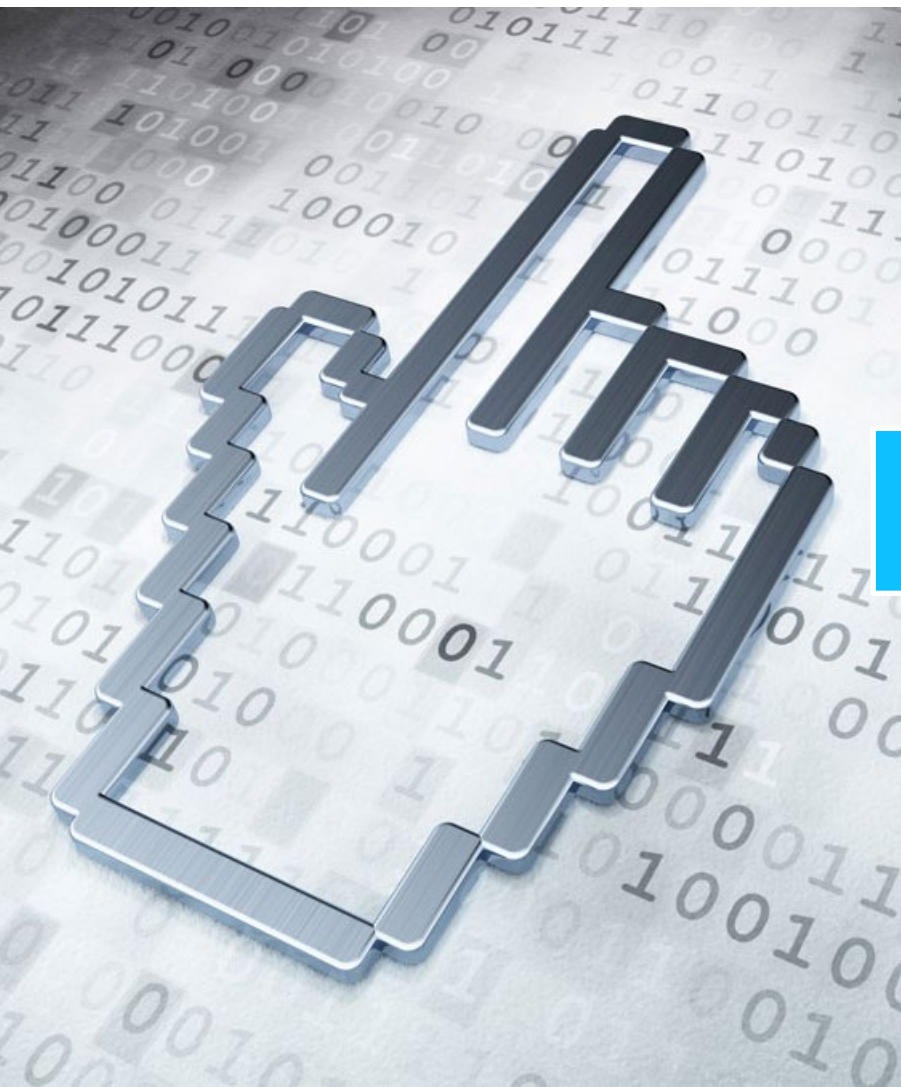
设计一个**S属性**的SDD来计算S.val，即输入串的十进制数值。比如，串101.101应该被翻译成十进制数5.625。

## 习题9.4

- 下面的SDT计算了一个由0和1组成的串的值。它把输入的符号串当作按照正二进制数来解释。

$$B \rightarrow B_1 0 \{B.val = 2 * B_1.val\}$$
$$| B_1 1 \{B.val = 2 * B_1.val + 1\}$$
$$| 1 \{B.val = 1\}$$

改写这个SDT，使得基础文法不再是左递归的，但仍然可以计算出整个输入串的相同的B.val的值。

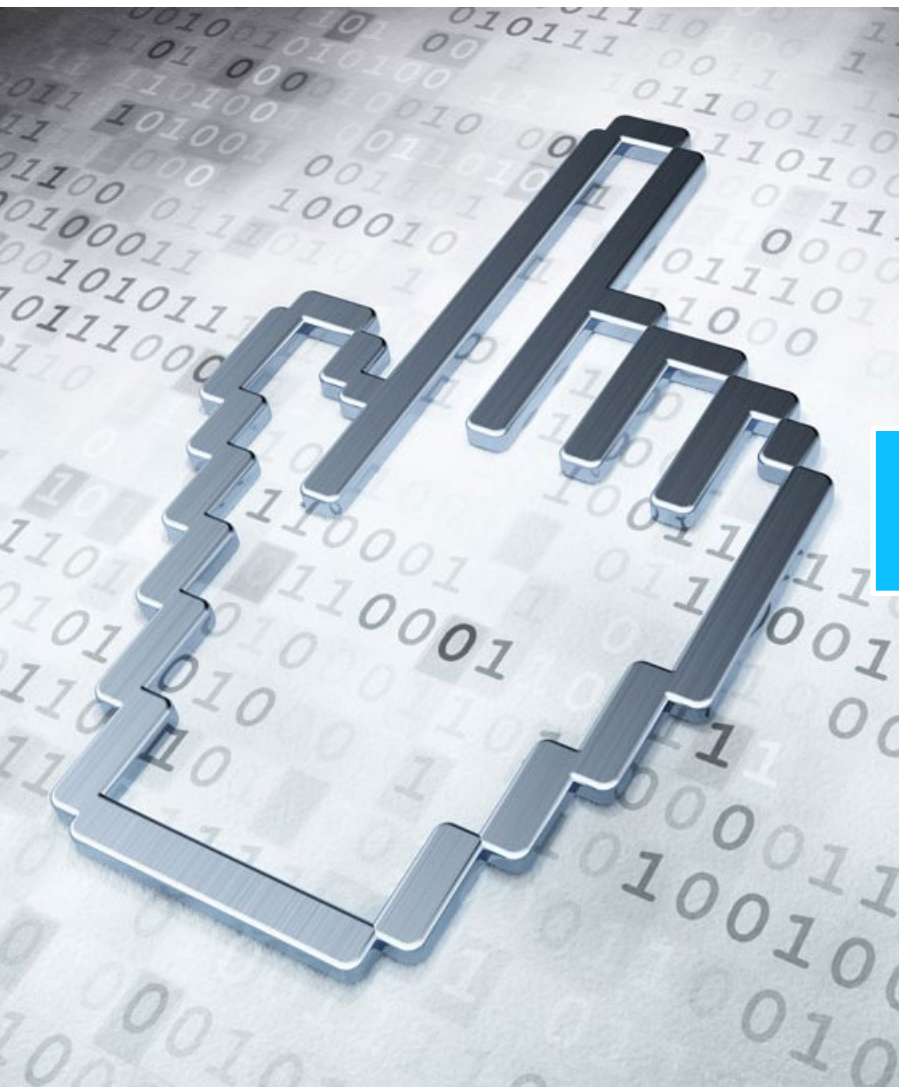


# 第10讲习题

## 习题10.1

- 将习题9.2中得到的SDD实现为递归下降的语法分析器。





# 第12讲习题

## 习题12.1

➤ 将算术表达式  $a + -(b + c)$  翻译成四元式序列

## 习题12.2

➤ 将下列赋值语句翻译成四元式序列（假设每个数组元素占8个存储单元）。

➤ (1)  $a = b[i] + c[j]$

➤ (2)  $a[i] = b * c - b * d$

➤ (3)  $x = f(y+1) + 2$

➤ (4)  $x = *p + \&y$

## 习题12.3

- 使用讲义中的翻译方案（如下图所示）翻译下列赋值语句。  
假设a和b的类型表达式都是 `array(3, array(5, real))`，数组c中每个元素（real类型）占8个存储单元

$S \rightarrow id = E;$

$| L = E; \{ gen( L.array \text{ '[' } L.offset \text{ ']' '}' E.addr ); \}$

$E \rightarrow E_1 + E_2 \mid -E_1 \mid (E_1) \mid id$

$| L \{ E.addr = newtemp(); gen( E.addr \text{ '=' } L.array \text{ '[' } L.offset \text{ ']' '}' ); \}$

$L \rightarrow id [E] \{ L.array = lookup(id.lexeme); if L.array == nil then error ;$

$L.type = L.array.type.elem ;$

$L.offset = newtemp();$

$gen( L.offset \text{ '=' } E.addr \text{ '*' } L.type.width ); \}$

$| L_1[E] \{ L.array = L_1.array;$

$L.type = L_1.type.elem ;$

$t = newtemp();$

$gen( t \text{ '=' } E.addr \text{ '*' } L.type.width );$

$L.offset = newtemp();$

$gen( L.offset \text{ '=' } L_1.offset \text{ '+' } t ); \}$

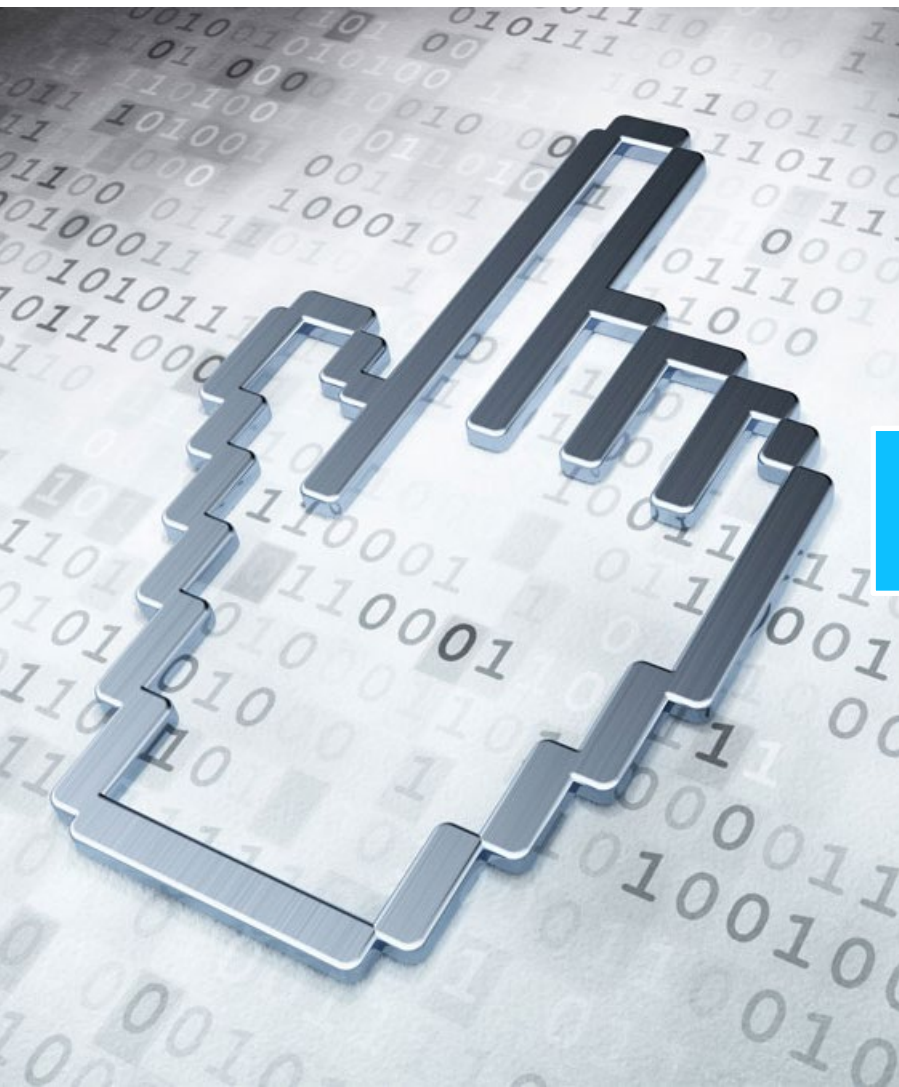
➤ (1)  $x = a[i][j] + b[i][j]$

➤ (2)  $x = a[b[i][j]][c[k]]$

## 习题12.4

- 修改下图中的翻译方案，使之适合Fortran风格的数组引用，也就是说， $n$ 维数组的引用为 $\text{id}[E_1, E_2, \dots, E_n]$ 。

```
 $S \rightarrow \text{id} = E;$   
  |  $L = E;$  {  $\text{gen}( L.\text{array} \text{'[' } L.\text{offset} \text{' '}' '=' E.\text{addr} );$  }  
 $E \rightarrow E_1 + E_2 \mid -E_1 \mid (E_1) \mid \text{id}$   
  |  $L$  {  $E.\text{addr} = \text{newtemp}(); \text{gen}( E.\text{addr} '=' L.\text{array} \text{'[' } L.\text{offset} \text{' '}' );$  }  
 $L \rightarrow \text{id} [E]$  {  $L.\text{array} = \text{lookup}(\text{id}.\text{lexeme});$  if  $L.\text{array} == \text{nil}$  then error ;  
                   $L.\text{type} = L.\text{array}.\text{type}.\text{elem} ;$   
                   $L.\text{offset} = \text{newtemp}();$   
                   $\text{gen}( L.\text{offset} '=' E.\text{addr} '*' L.\text{type}.\text{width} );$  }  
  |  $L_1[E]$  {  $L.\text{array} = L_1.\text{array};$   
               $L.\text{type} = L_1.\text{type}.\text{elem} ;$   
               $t = \text{newtemp}();$   
               $\text{gen}( t '=' E.\text{addr} '*' L.\text{type}.\text{width} );$   
               $L.\text{offset} = \text{newtemp}();$   
               $\text{gen}( L.\text{offset} '=' L_1.\text{offset} '+' t );$  }
```



# 第13讲习题

## 习题13.1

- 在SPOC讲义6.3节（控制语句的翻译）所示的SDT中添加处理下列控制流构造的翻译方案
  - (1) repeat语句:  $S \rightarrow \text{repeat } S_1 \text{ while } B$
  - (2) for循环语句:  $S \rightarrow \text{for } (S_1 ; B ; S_2 ) S_3$

## 习题13.2

- 为下面的产生式写出一个和SPOC讲义6.3节（控制语句的翻译）中SDT类似的SDT。该产生式表示一个常见的C语言中的控制流结构。

$$S \rightarrow '\{ ' L '\}'$$

$$L \rightarrow L S \mid \varepsilon$$



## 习题13.3(1)

➤ 试将下面的语句翻译成四元式序列

(1) while  $a < c \wedge b < d$  do

    if  $a = 1$  then  $c := c + 1$

    else while  $a \leq d$  do

$a := a + 2$

## 习题13.3(2)

➤ 试将下面的语句翻译成四元式序列

(2) **for i:=m step 2 until n do**

**if a<b then x:=x+1**

## 习题13.3(3)

➤ 试将下面的语句翻译成四元式序列

(3) **for**  $i:=1$  **step** 1 **until**  $n$  **do**

**while**  $a < b$  **do**

**if**  $c > d$  **then**  $x := -b + c$  **else**  $x := a * b + c$

## 习题13.3(4)

➤ 试将下面的语句翻译成四元式序列

(4) if  $w < 1$  then  $a := b * c + d$  else repeat  $a := a - 1$  until  $a < 0$

## 习题13.4

➤ Pascal语言的标准将for语句

**for v := E1 to E2 do S1**

定义成和下面的代码序列有同样的含义:

**begin**

**v := E1;**

**while v <= E2 do begin**

**S1;**

**v := succ(v);**

**end**

**end**

请为for语句设计一种合理的中间代码结构，并写出产生中间代码的翻译方案

## 习题13.5

➤ 设有如下的文法G:

$S \rightarrow \text{id} := E$

| if B then S

| while B do S

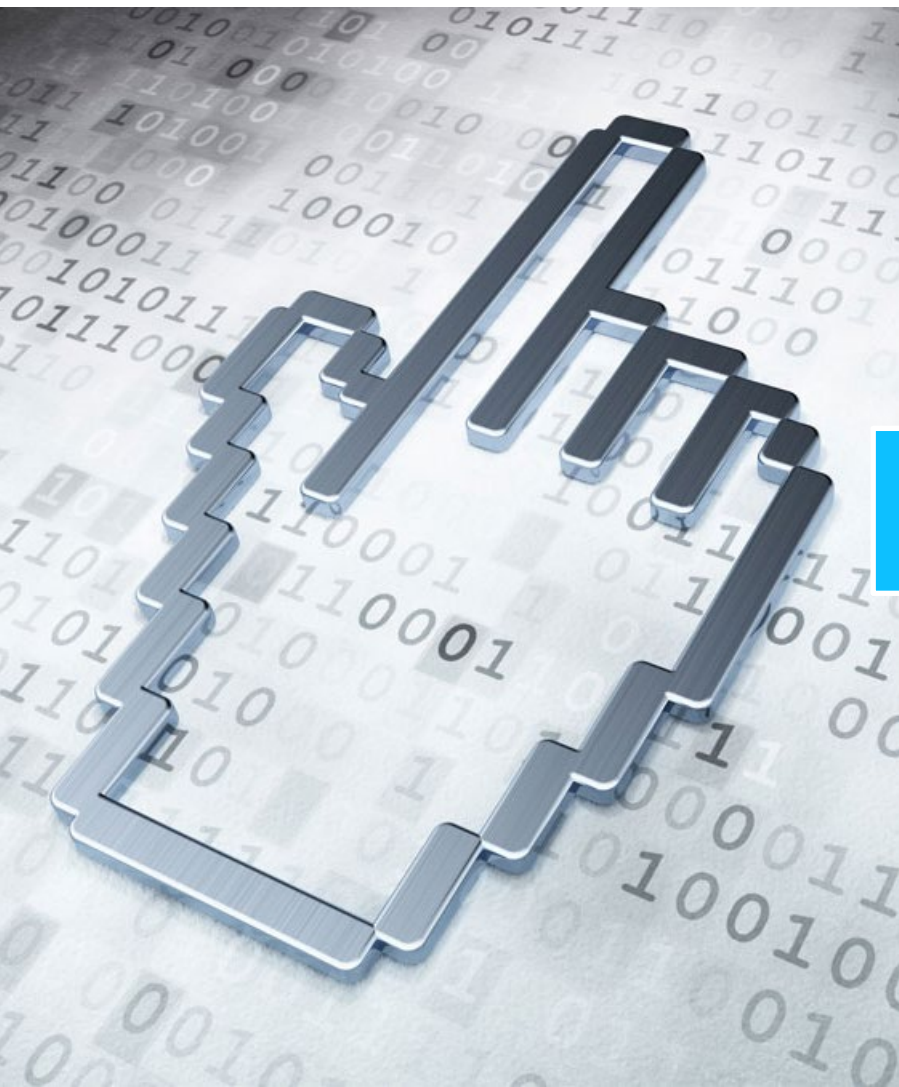
| begin S ; S end

| break

试给出完成如下翻译所要求的翻译模式：如果发现break未出现在循环语句中，则报告错误。

## 习题13.6

- 写出条件赋值语句  $\text{id} := \text{if } B \text{ then } E_1 \text{ else } E_2$  的语义子程序。其中， $B$  为布尔表达式， $E_1$  和  $E_2$  是算术表达式， $i$  代表与  $E_1$  和  $E_2$  同类的简单变量。按照写出的语义子程序，生成条件赋值语句  $m := \text{if } a > c \text{ then } x + y \text{ else } x - y + 0.5$  的四元式序列



# 第14讲习题



## 习题14.1

➤使用SPOC讲义6.4节（回填）中的翻译方案翻译下列表达式。给出每个子表达式的truelist和falselist。你可以假设第一条被生成的指令的地址是100。

➤(1)  $a == b \ \&\& \ (c == d \ || \ e == f)$

➤(2)  $(a == b \ || \ c == d) \ || \ e == f$

➤(3)  $(a == b \ \&\& \ c == d) \ \&\& \ e == f$

## 习题14.2

➤ 下图a中给出了一个程序的摘要。b概述了使用SPOC讲义6.4节（回填）中的回填翻译方案生成的三地址代码的结构。这里， $i_1 \sim i_8$ 是每个code区域的第一条被生成的指令的标号。请给出最终回填到下列列表中的标号（即 $i_1 \sim i_8$ 中的某个标号）。

- (1)  $E_3.\text{falselist}$
- (2)  $S_2.\text{nextlist}$
- (3)  $E_4.\text{falselist}$
- (4)  $S_1.\text{nextlist}$
- (5)  $E_2.\text{truelist}$

```
while ( $E_1$ ) {  
    if ( $E_2$ )  
        while ( $E_3$ )  
             $S_1$ ;  
    else {  
        if ( $E_4$ )  
             $S_2$ ;  
         $S_3$   
    }  
}
```

(a)

$i_1$ : Code for  $E_1$   
 $i_2$ : Code for  $E_2$   
 $i_3$ : Code for  $E_3$   
 $i_4$ : Code for  $S_1$   
 $i_5$ : Code for  $E_4$   
 $i_6$ : Code for  $S_2$   
 $i_7$ : Code for  $S_3$   
 $i_8$ : ...

(b)

## 习题14.3

- 使用SPOC讲义6.4节（回填）中的翻译方案对下图进行翻译时，我们为每条语句创建S.nextlist列表。一开始是赋值语句 $S_1$ 、 $S_2$ 、 $S_3$ ，然后逐步处理越来越大的if语句、while语句和语句块。在下图中有5个这种类型的结构语句：

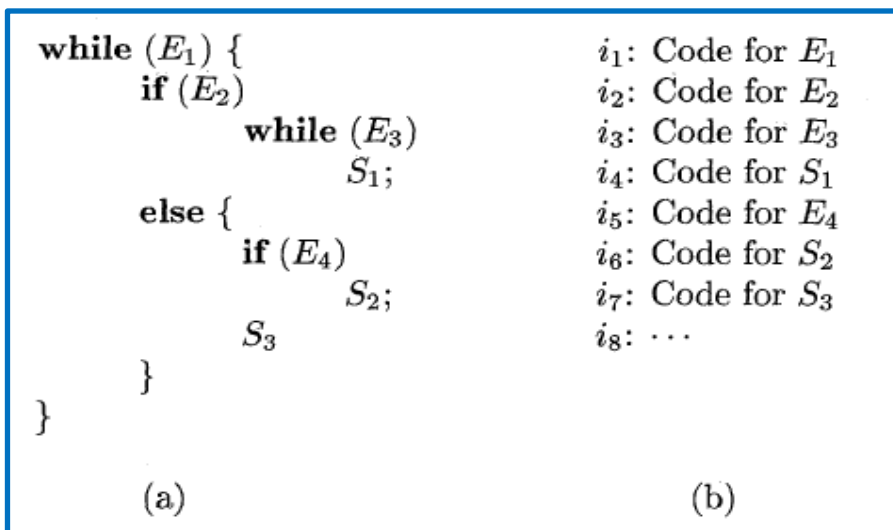
$S_4$ : while ( $E_3$ )  $S_1$

$S_5$ : if ( $E_4$ )  $S_2$

$S_6$ : 包含 $S_5$ 和 $S_3$ 的语句块

$S_7$ : 语句if ( $E_2$ )  $S_4$  else  $S_6$

$S_8$ : 整个程序



对于这些结构语句，我们可以通过一个规则用其他的 $S_j$ .nextlist列表以及程序中的表达式的列表 $E_k$ .truelist和 $E_k$ .falselist构造出 $S_i$ .nextlist。给出计算下列nextlist列表的规则：

- (1)  $S_4$ .nextlist
- (2)  $S_5$ .nextlist
- (3)  $S_6$ .nextlist
- (4)  $S_7$ .nextlist
- (5)  $S_8$ .nextlist

## 习题14.4

- 在SPOC讲义6.4节（回填）所示的SDT中添加处理如下for循环语句的翻译方案

$$S \rightarrow \textit{for} (S_1 ; B ; S_2 ) S_3$$

## 习题14.5

- 设有形如  $\text{call}(E) P_1, P_2, P_3$  的过程调用语句，其中， $E$  为算术表达式， $P_1$ 、 $P_2$  和  $P_3$  是3个不带参数的过程的名字。该调用语句执行的流程为：首先计算表达式  $E$  的值， $E=0$  时调用  $P_1$ ， $E>0$  时调用  $P_2$ ， $E<0$  时调用  $P_3$ 。该语句执行时， $P_1$ 、 $P_2$  和  $P_3$  中有且仅有一个被调用。
  - (1) 试给出该调用语句的三地址码。
  - (2) 给出该调用语句适于自底向上分析的SDT

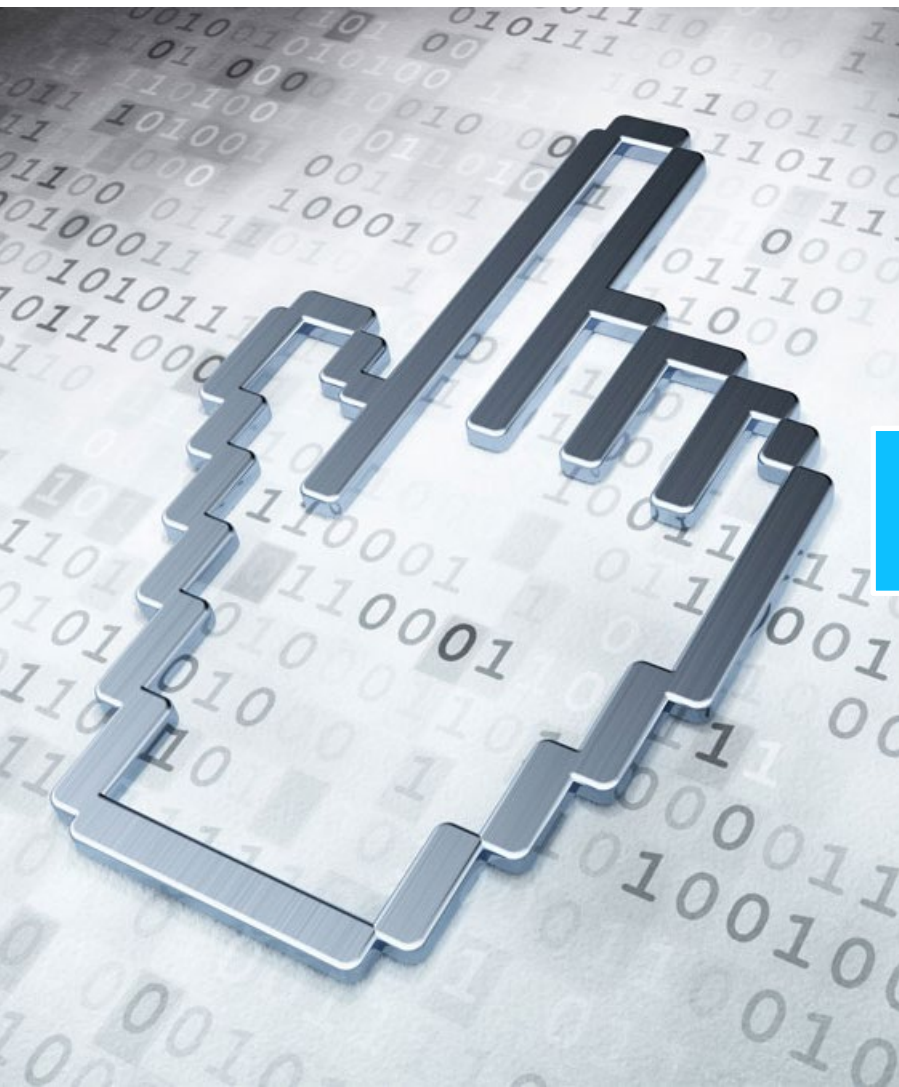
## 习题14.6

➤ 给定如下文法：

$$S \rightarrow (L) \mid a$$

$$L \rightarrow L, S \mid S$$

- (1) 试设计一个SDD，输出S所生成的串中的配对括号的个数。
- (2) 试设计一个SDT，输出S所生成的串中每个a的嵌套深度。例如，串(a,(a,a))的输出结果是1、2、2。



# 第15讲习题

## 习题15.1

- 假设SPOC讲义7.1节（栈式存储分配）中的程序（如下图所示）使用如下的partition函数：该函数总是将 $a[m]$ 作为分割值 $v$ 。同时假设在对数组 $a[m], \dots, a[n]$ 重新排序时总是尽量保存原来的顺序。也就是说，首先是以原顺序保持所有小于 $v$ 的元素，然后保存所有等于 $v$ 的元素，最后按原来顺序保存所有大于 $v$ 的元素。
- (1) 画出对数字9、8、7、6、5、4、3、2、1进行排序时的活动树。
- (2) 同时在栈中出现的活动记录最多有多少个？



# 习题15.1 (con.)

## 例：一个快速排序程序的概要

```
int a[11];
void readArray() /*将9个整数读入到a[1],...,a[9]中 */
{
    int i;
    ...
}
int partition(int m, int n)
{
    /*选择一个分割值v, 划分a[m...n], 使得a[m...p-1]小于v, a[p]=v,
      a[p+1...n]大于等于v。返回p */
    ...
}
void quicksort(int m, int n)
{
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main()
{
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1, 9);
}
```

## 习题15.2

- 当初始顺序为1、3、5、7、9、2、4、6、8时，重复习题15.1

## 习题15.3

➤ 下图是递归计算Fibonacci数列的C语言代码。假设f的活动记录按顺序包含下列元素：（返回值，参数n，局部变量s，局部变量t）。通常在活动记录中还会有其他元素。下面的问题假设初始调用是f(5)。

- (1) 给出完整的活动树。
- (2) 当第1个f(1)调用即将返回时，运行时刻栈和其中的活动记录是什么样子的？
- (3) 当第5个f(1)调用即将返回时，运行时刻栈和其中的活动记录是什么样子的？

```
int f(int n) {  
    int t, s;  
    if (n < 2) return 1;  
    s = f(n-1);  
    t = f(n-2);  
    return s+t;  
}
```

## 习题15.4

➤ 下面是两个C语言函数f和g的概述：

```
int f ( int x) { int i ; ...return i+1; ...}
```

```
int g ( int y) { int j; ... f ( j+1 ) ...}
```

也就是说，函数g调用函数f。画出在g调用f而f即将返回时，运行时刻栈中从g的活动记录开始的顶端部分。你可以只考虑返回值、参数、控制链以及存放局部数据的空间。你不用考虑存放的机器状态，也不用考虑没有在代码中显示的局部值和临时值。但是你应该指出：

- (1) 哪个函数在栈中为各个元素创建了所使用的空间？
- (2) 哪个函数写入了各个元素的值？
- (3) 这些元素属于哪个活动记录？

## 习题15.5

- 下图中给出了一个按照非标准方式计算Fibonacci数的ML语言的函数main。函数fib0将计算第n个Fibonacci数 ( $n \geq 0$ )。嵌套在fib0中的是fib1，它假设  $n \geq 2$  并计算第n个Fibonacci。嵌套在fib1中的是fib2，它假设  $n \geq 4$ 。请注意，fib1和fib2都不需要检查基本情况。我们考虑从对main的调用开始，直到（对fib0(1)的）第一次调用即将返回的阶段。请描述出当时的活动记录栈，并给出栈中的各个活动记录的访问链。

## 习题15.5 (con.)

```
fun main () {  
  let  
    fun fib0(n) =  
      let  
        fun fib1(n) =  
          let  
            fun fib2(n) = fib1(n-1) + fib1(n-2)  
          in  
            if n >= 4 then fib2(n)  
            else fib0(n-1) + fib0(n-2)  
          end  
        in  
          if n >= 2 then fib1(n)  
          else 1  
        end  
      in  
        fib0(4)  
      end;  
}
```

## 习题15.6

- 在本讲中，我们提到，编译器通常为每个作用域（程序块）建立一个独立的符号表（SPOC讲义p49）。有关作用域和块结构的概念参见教材1.6.1节（静态和动态的区别）和1.6.3节（静态作用域和块结构）。对于下图所示的块结构代码（为便于引用各语句，增加了行号），假设使用常见的声明的静态作用域规则，请完成以下习题。

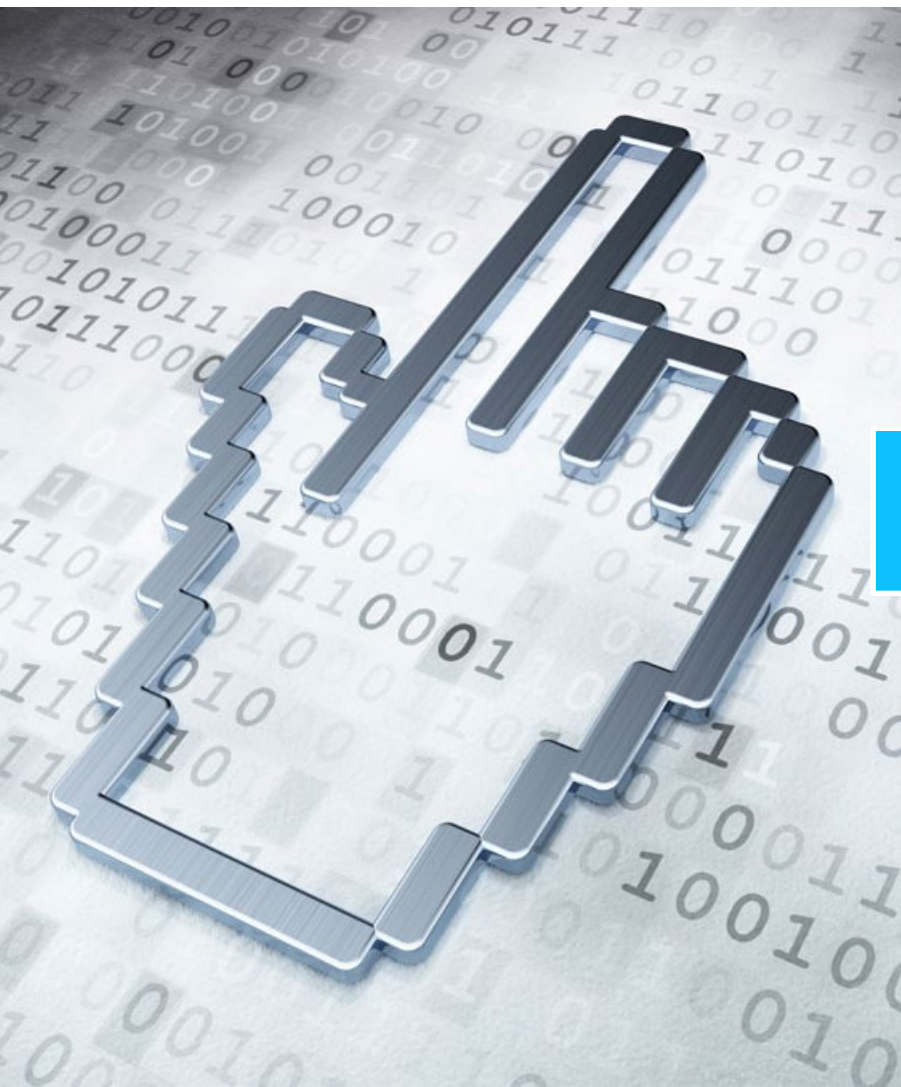
```
(1)  {    int w, x, y, z;          /* Block B1 */
(2)      {    int x, z;           /* Block B2 */
(3)          {    int w, x;      /* Block B3 */ }
(4)      }
(5)  {    int w, x;               /* Block B4 */
(6)      {    int y, z;          /* Block B5 */ }
(7)      }
(8)  }
```



## 习题15.6 (con.)

- (1) 判断下列说法的对错
  - B1声明的w的作用域是1-8行
  - B1声明的y的作用域是1-5行
- (2) 给出该代码片段中12个声明中的每一个的作用域
- (3) 画出该代码片段的符号表





# 第16讲习题

## 习题16.1

➤ 下图是一个简单的矩阵乘法程序。

```
for (i=0; i<n; i++)  
    for (j=0; j<n; j++)  
        c[i][j] = 0.0;  
for (i=0; i<n; i++)  
    for (j=0; j<n; j++)  
        for (k=0; k<n; k++)  
            c[i][j] = c[i][j] + a[i][k]*b[k][j];
```

- (1) 假设矩阵的元素是需要8个字节的数值，而且矩阵按行存放。把程序翻译成三地址语句。
- (2) 为(1)中得到的代码构造流图。
- (3) 找出在(2)中得到的流图的循环。

## 习题16.2

➤下图中是计算从2~n之间素数个数的代码。

```
for (i=2; i<=n; i++)
    a[i] = TRUE;
count = 0;
s = sqrt(n);
for (i=2; i<=s; i++)
    if (a[i]) /* 已知i是一个素数 */ {
        count++;
        for (j=2*i; j<=n; j = j+i)
            a[j] = FALSE; /* i的倍数都不是素数 */
    }
```

- (1) 假设数组的元素是需要4个字节存放的的整数，把程序翻译成三地址语句。
- (2) 为(1)中得到的代码构造流图。
- (3) 找出在(2)中得到的流图的循环。

## 习题16.3

➤ 为下面的表达式构造DAG

$$((x + y) - ((x + y) * (x - y))) + ((x + y) * (x - y))$$

## 习题16.4

➤ 为下列表达式构造DAG。假定+是左结合的。

➤ (1)  $a + b + (a + b)$

➤ (2)  $a + b + a + b$

➤ (3)  $a + a + ((a + a + a + (a + a + a + a)) )$

## 习题16.5

➤ 对于下面的基本块

$$d = b * c$$

$$e = a + b$$

$$b = b * c$$

$$a = e - d$$

➤ (1) 为该基本块构造DAG

➤ (2) 分别按照下列两种假设简化上述三地址代码

➤ 只有a在基本块的出口活跃

➤ a、b、c在基本块的出口活跃

## 习题16.6

➤ 为下面的基本块构造DAG。请不要忘记包含比较指令  $i \leq 10$

$t5 = i - 1$

$t6 = 88 * t5$

$a[t6] = 1.0$

$i = i + 1$

if  $i \leq 10$  goto B6

## 习题16.7

➤ 为下面的基本块构造DAG。

$t1 = 10 * i$

$t2 = t1 + j$

$t3 = 8 * t2$

$t4 = t3 - 88$

$a[t4] = 0.0$

$j = j + 1$

if  $j \leq 10$  goto B3

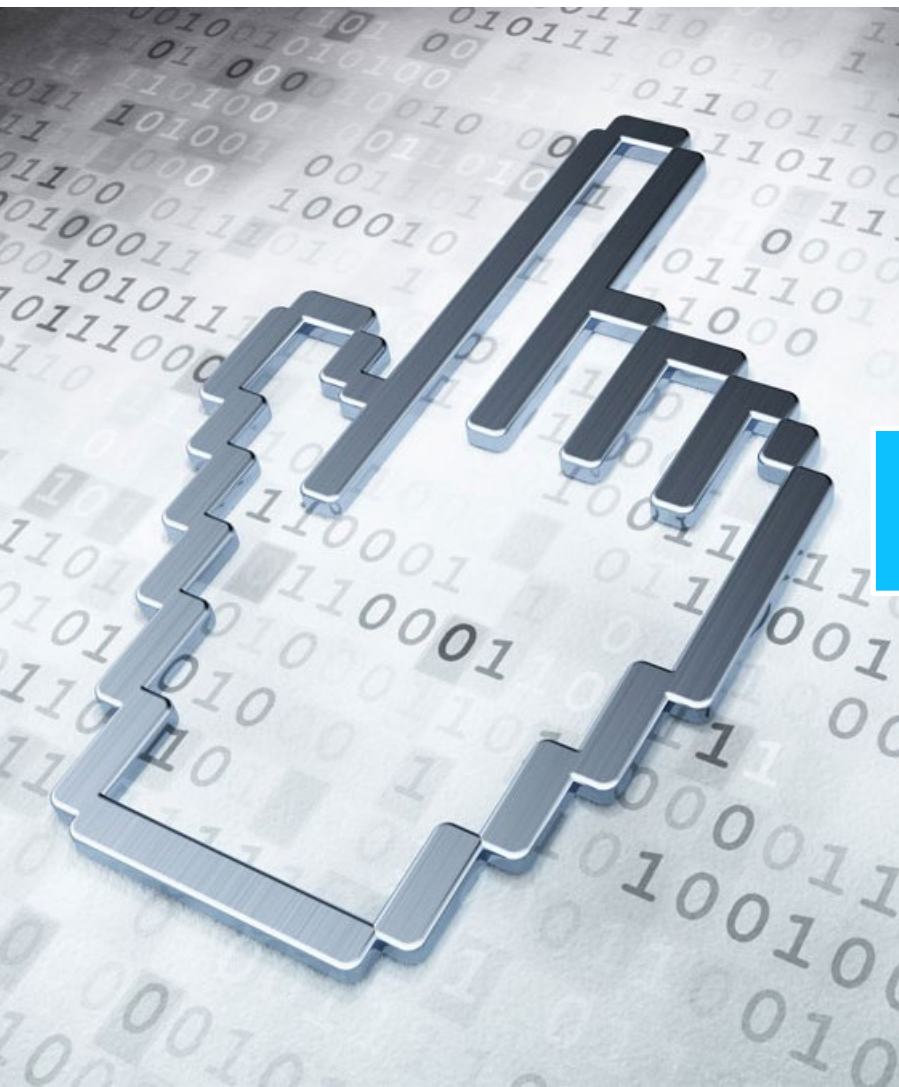


## 习题16.8

➤ 考虑下面的三地址语句序列：

```
    b = 1
    b = 2
    if w ≤ x goto L2
    e = b
    goto L2
L1:  goto L3
L2:  c = 3
     b = 4
     c = 6
L3:  if y ≤ z goto L4
     goto L5
L4:  g = g + 1
     h = 8
     goto L1
L5:  h = 9
```

- (1) 在该代码中用横线将代码分成基本块，并给每个基本块赋一个序号
- (2) 画出该代码的流图，每个基本块就用(1)中的序号表示
- (3) 若有循环，则列出构成每个循环的结点



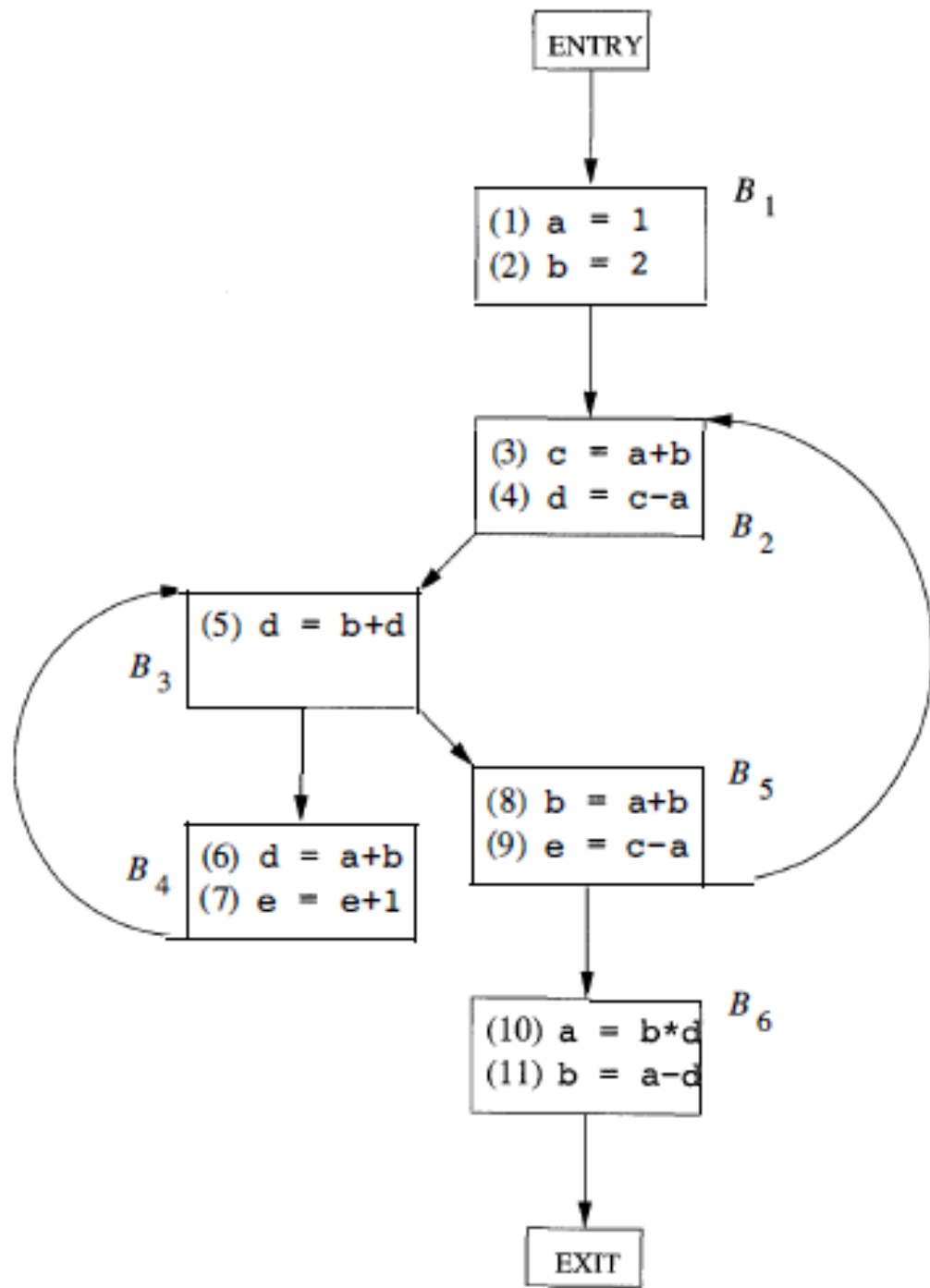
# 第17讲习题

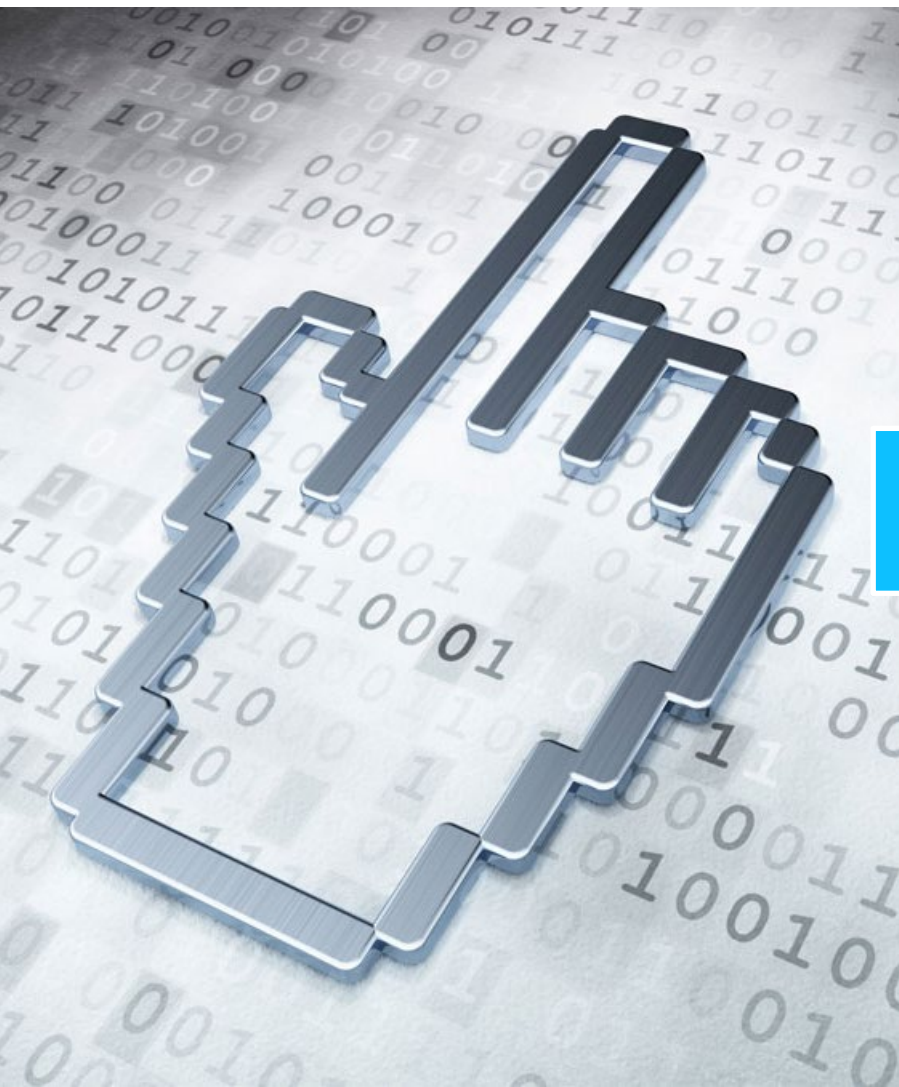
# 习题17.1

➤ 对下图中的流图，  
计算下列值：

➤ (1) 每个基本块的  
gen和kill集合。

➤ (2) 每个基本块的IN  
和OUT集合。

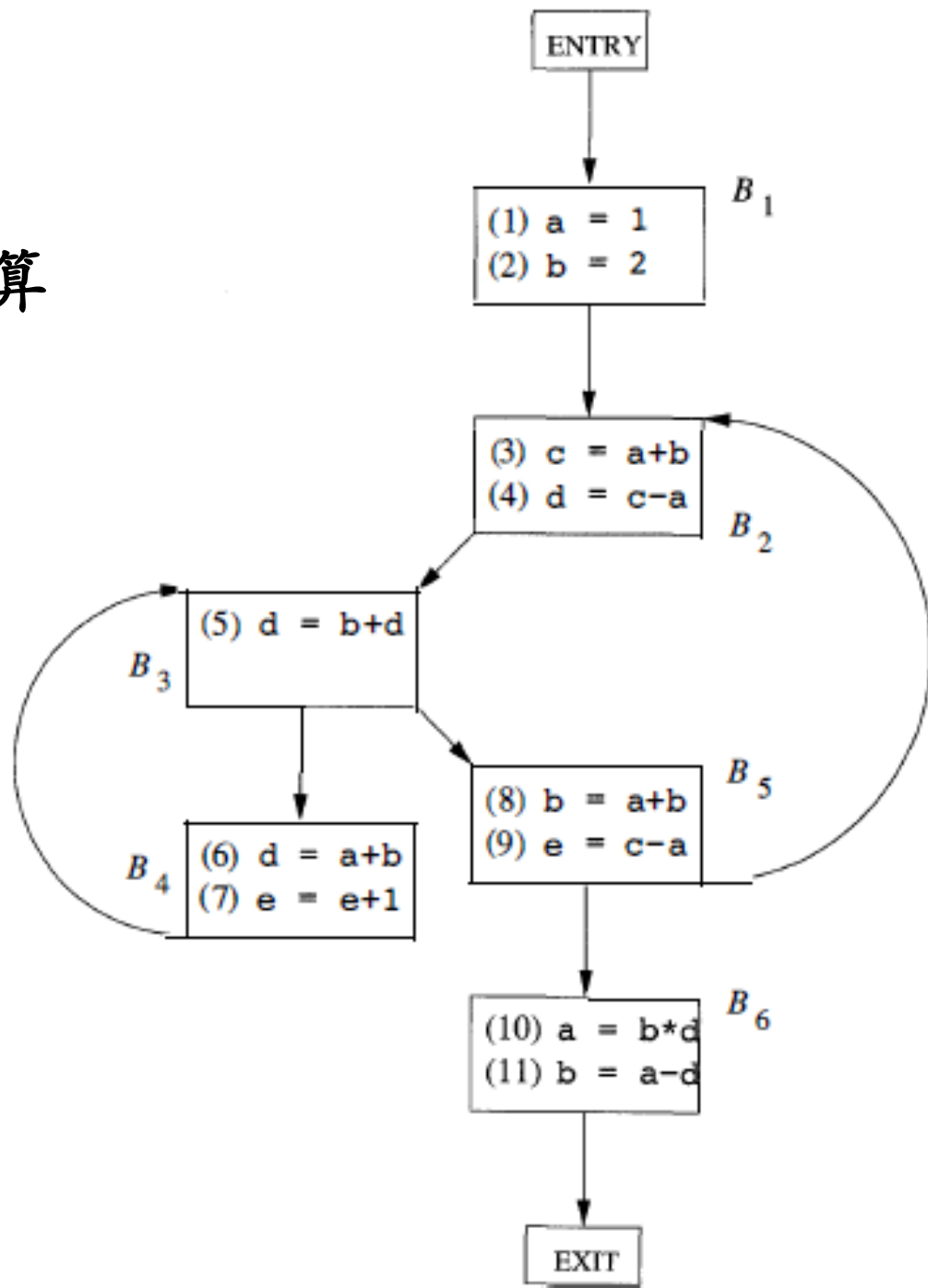




# 第18讲习题

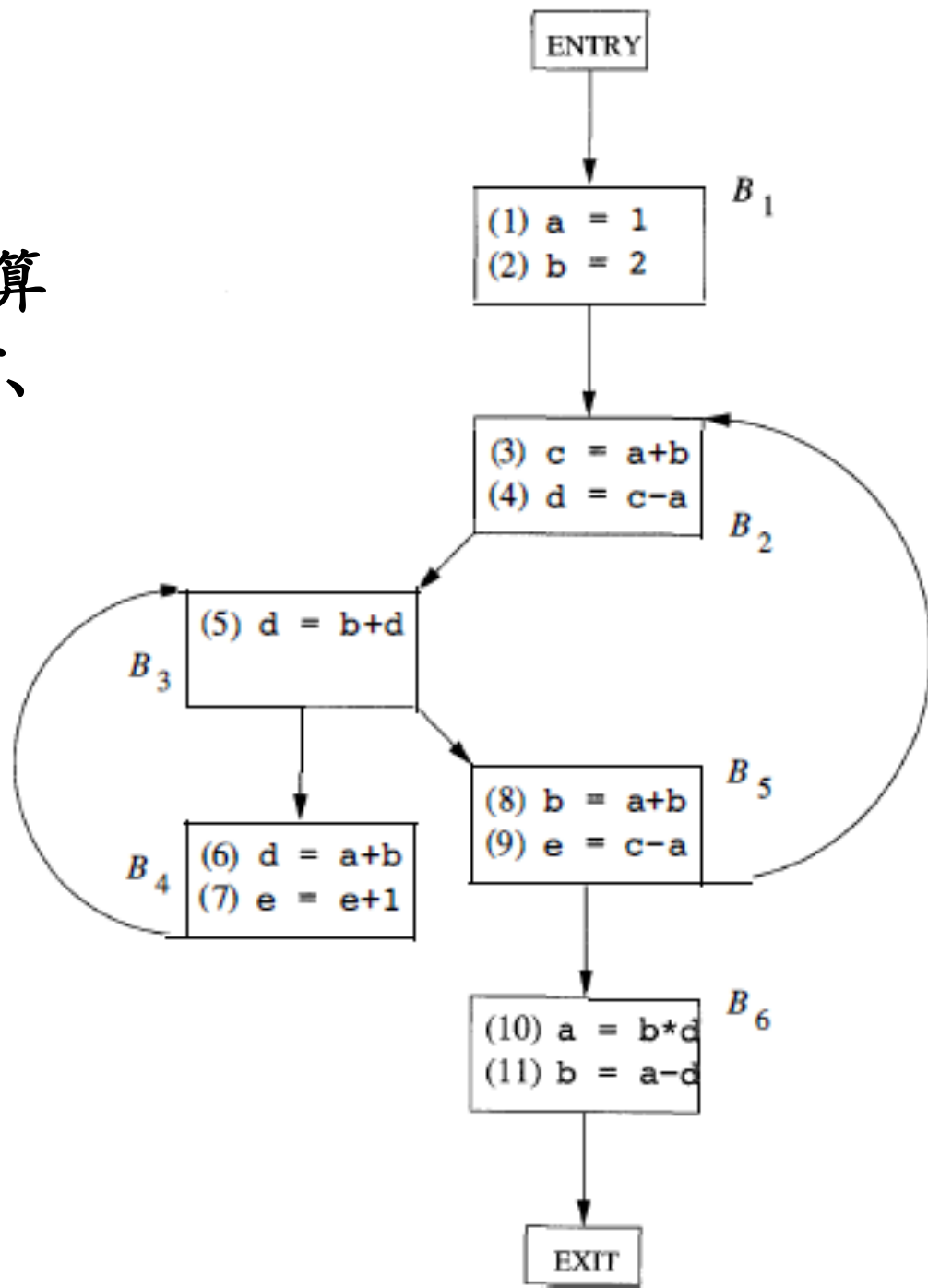
## 习题18.1

➤ 对下图中的流图，计算可用表达式问题中的  $e\_gen$ 、 $e\_kill$ 、IN 和 OUT 集合。



## 习题18.2

- 对下图中的流图，计算活跃变量分析中的def、use、IN和OUT集合。



## 习题18.3

- 程序中的变量，若在第1次引用前没有置初值的话，则称它为未初始化变量。请运用数据流分析技术，给出计算程序的未初始化变量集合的方法。

## 习题18.4

- 考虑一个类Pascal的语言，其中所有的变量都是整型（不需要显式声明），并且仅包含赋值语句、读语句、写语句、条件语句和循环语句。下面的产生式定义了该语言的语法（其中lit表示整型常量；OP的产生式没有给出，因为它和下面讨论的问题无关）

$Program \rightarrow Stmt$

$Stmt \rightarrow id := Exp$

$Stmt \rightarrow read ( id )$

$Stmt \rightarrow write( Exp )$

$Stmt \rightarrow Stmt ; Stmt$

$Stmt \rightarrow if (Exp) then begin Stmt end$   
 $else begin Stmt end$

$Stmt \rightarrow while (Exp) do begin Stmt end$

$Exp \rightarrow id$

$Exp \rightarrow lit$

$Exp \rightarrow Exp OP Exp$

定义*Stmt*的两个属性：*MayDef*表示它可能定值的变量集合，*MayUse*表示它可能引用的变量集合。

- (1) 写一个SDD或SDT，它计算*Stmt*的*MayDef*属性和*MayUse*属性
- (2) 基于*MayDef*和*MayUse*属性，说明*Stmt*<sub>1</sub>；*Stmt*<sub>2</sub>和*Stmt*<sub>2</sub>；*Stmt*<sub>1</sub>在什么情况下有同样的语义



## 习题18.5

- 考虑一个类Pascal的语言，其中所有的变量都是整型（不需要显式声明），并且仅包含赋值语句、读语句、写语句、条件语句和循环语句。下面的产生式定义了该语言的语法（其中lit表示整型常量；OP的产生式没有给出，因为它和下面讨论的问题无关）

$Program \rightarrow Stmt$

$Stmt \rightarrow id := Exp$

$Stmt \rightarrow read ( id )$

$Stmt \rightarrow write( Exp )$

$Stmt \rightarrow Stmt ; Stmt$

$Stmt \rightarrow if (Exp) then begin Stmt end$   
 $else begin Stmt end$

$Stmt \rightarrow while (Exp) do begin Stmt end$

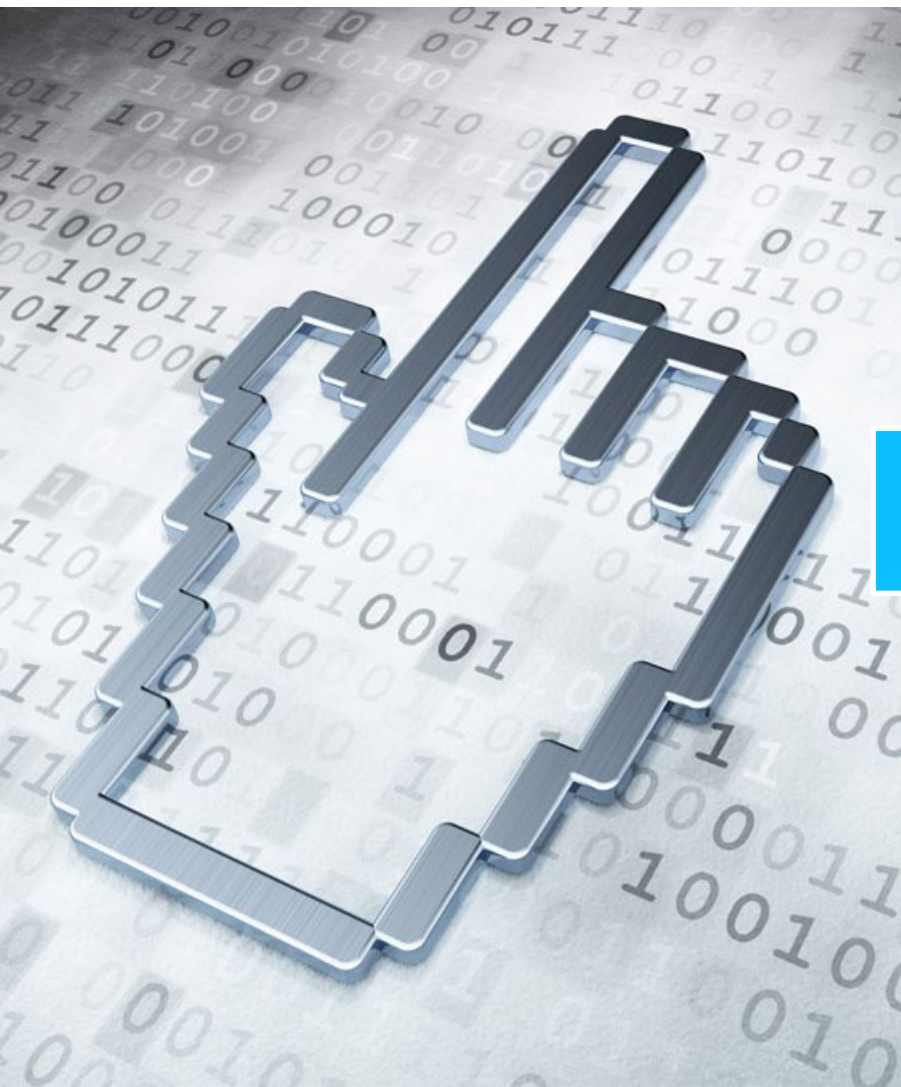
$Exp \rightarrow id$

$Exp \rightarrow lit$

$Exp \rightarrow Exp OP Exp$

定义*Stmt*的两个属性：*Def*表示*Stmt*中一定会定值且该定值前未引用的变量集合，*MayUse*表示*Stmt*中可能出现的引用前未定值的变量集合。

- (1) 写一个SDD或SDT，它计算*Stmt*的*Def*属性和*MayUse*属性
- (2) 基于上面的计算，程序可能未赋初值的变量集合从哪儿可以得到



# 第19讲习题

## 习题19.1

➤ 对于下图中的流图：

➤ (1) 找出流图中的循环。

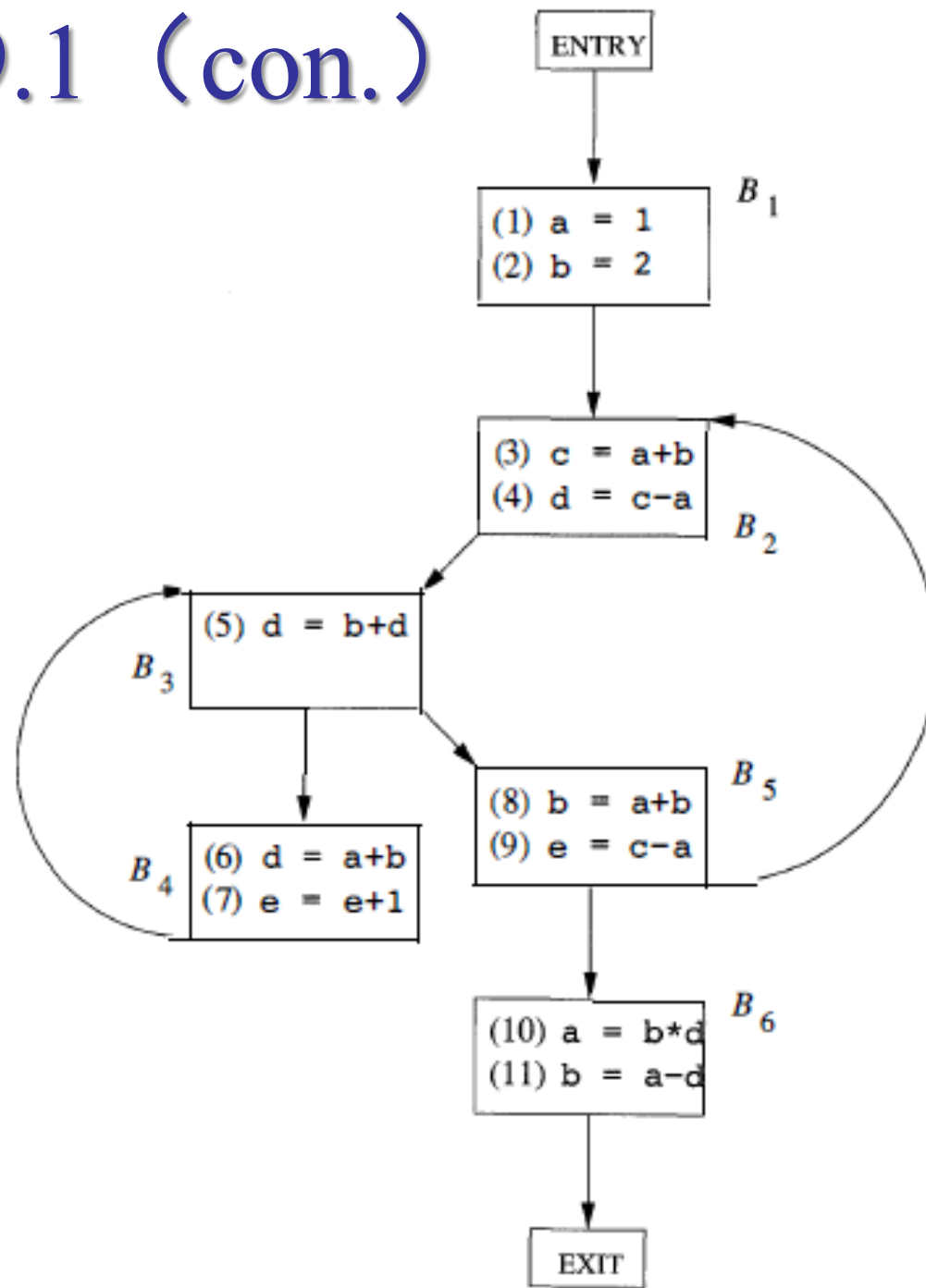
➤ (2) B1中的语句 (1) 和 (2) 都是复制语句。其中a和b都被赋予了常量值。我们可以对a和b的哪些使用进行复制传播，并把对它们的使用替换为对一个常量的使用？在所有可能的地方进行这种替换。

➤ (3) 对每个循环，找出所有的全局公公子表达式。

➤ (4) 寻找每个循环中的归纳变量。同时要考虑在 (2) 中引入的所有常量。

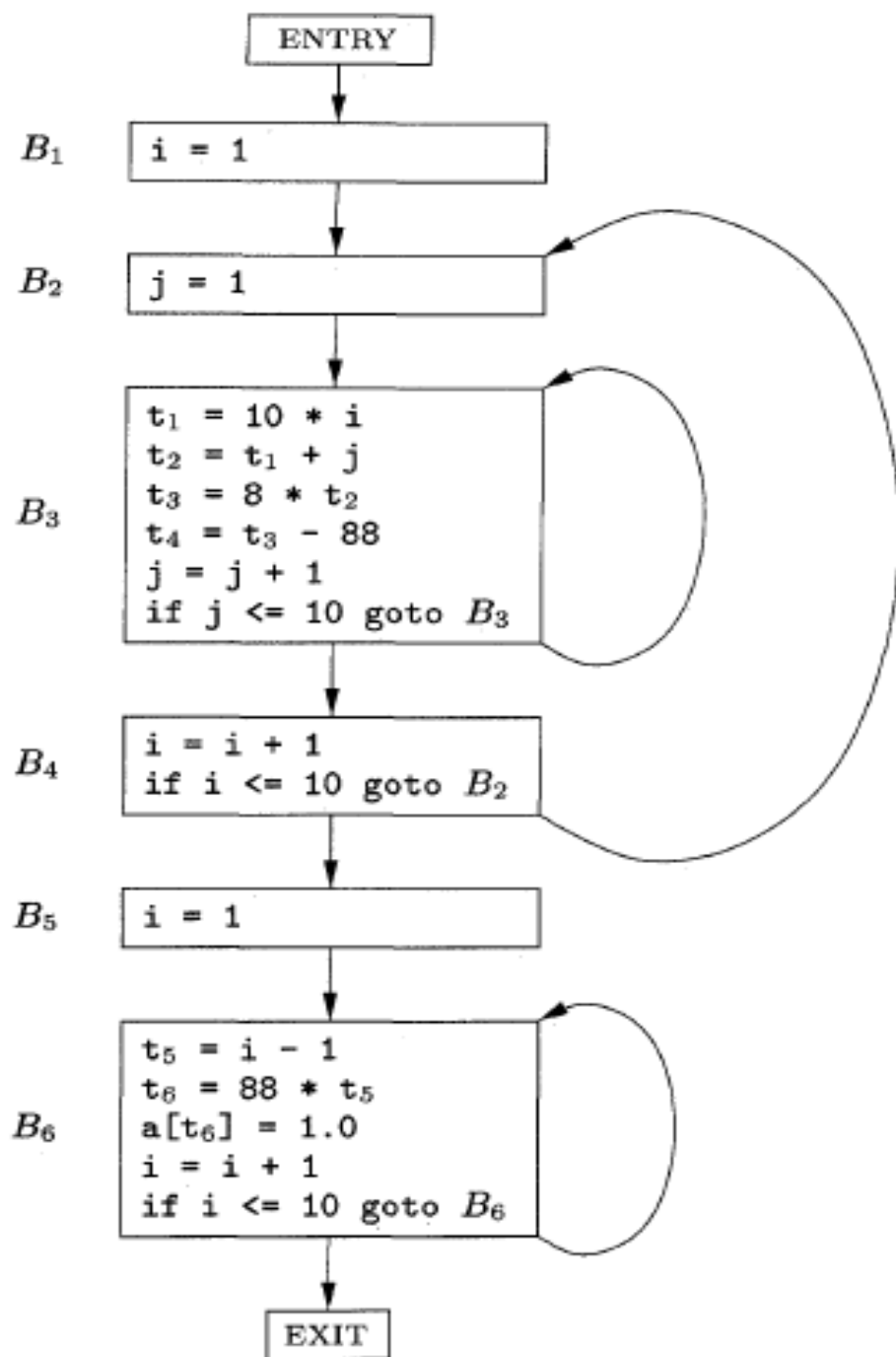
➤ (5) 寻找每个循环的全部循环不变计算。

# 习题19.1 (con.)



## 习题19.2

➤ 对于下图中的流图进行代码优化。



## 习题19.3

➤ 对习题16.1中得到的流图进行代码优化。

## 习题19.4

➤ 对习题16.2中得到的流图进行代码优化。

## 习题19.5

- 下图是用来计算两个向量A和B的点积的中间代码。尽你所能，通过下列方法优化这个代码：消除公共子表达式，对归纳变量进行强度消减，消除归纳变量。

dp 0.

i = 0

L: t1 = i \* 8

t2 = A[t1]

t3 = i \* 8

t4 = B[t3]

t5 = t2 \* t4

dp = dp + t5

i = i + 1

if i < n goto L



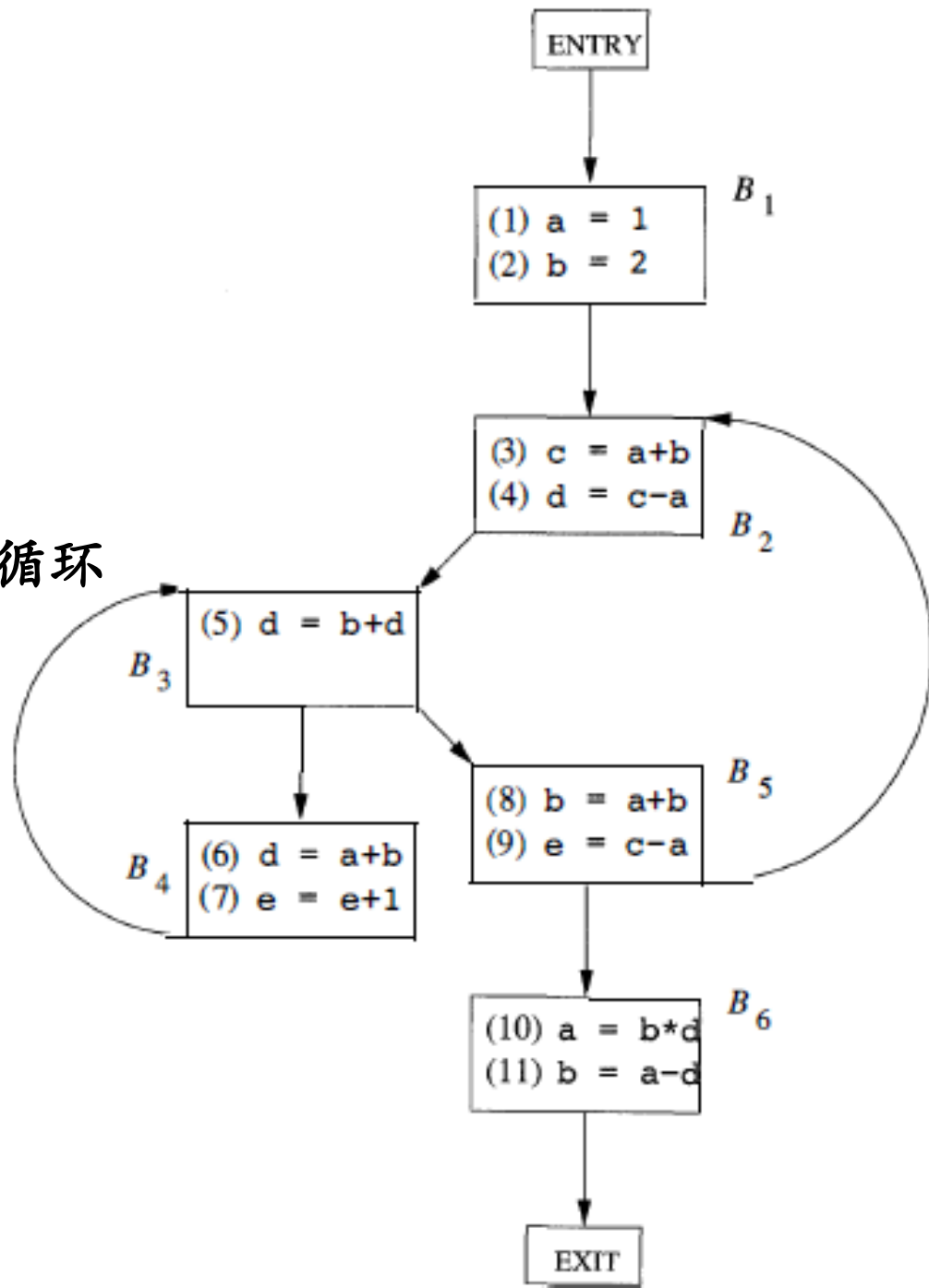
## 习题19.6

➤ 对下图中的流图：

➤ (1) 计算支配关系

➤ (2) 构造支配节点树

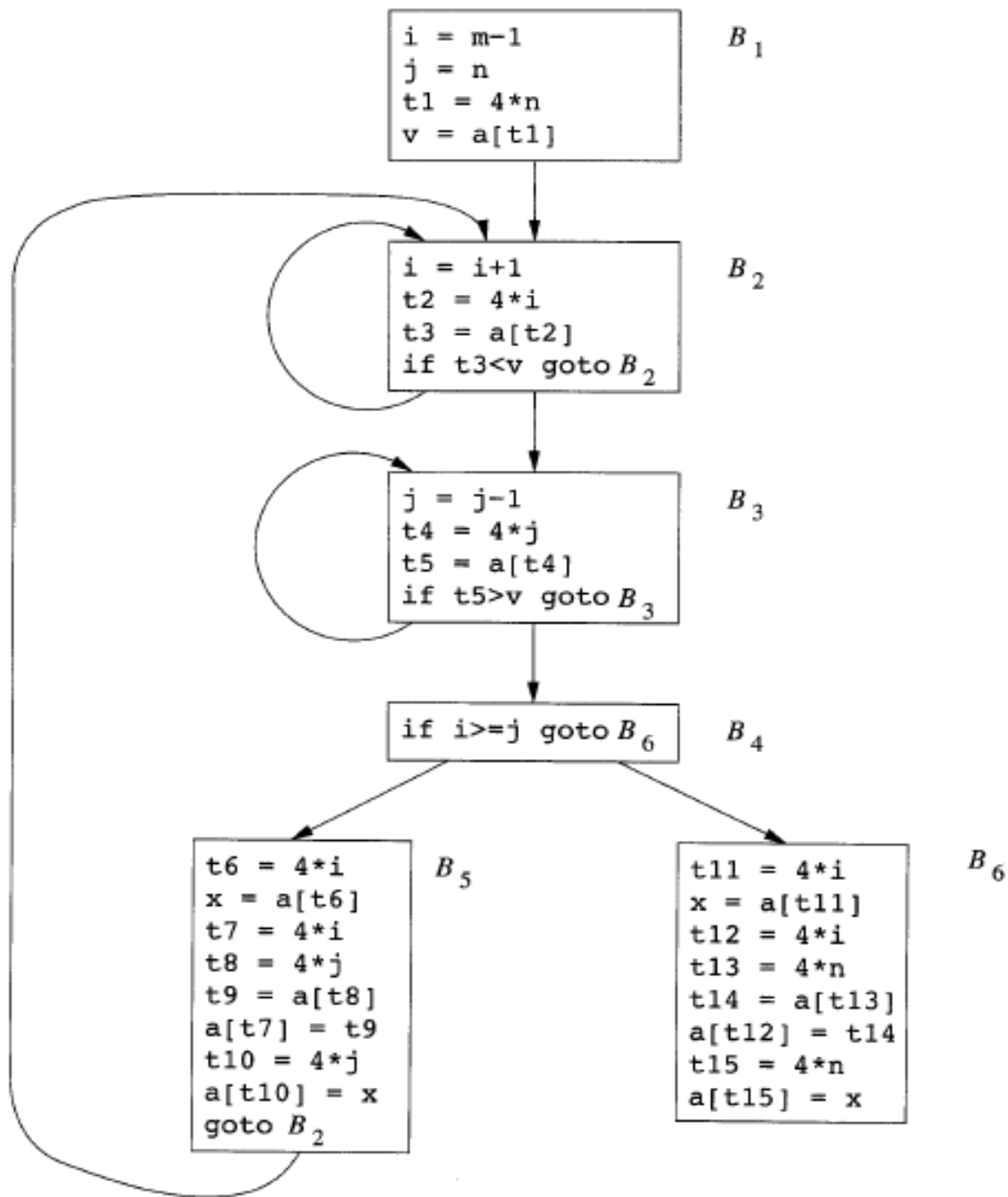
➤ (3) 找出这个流图的自然循环



## 习题19.7

➤ 对下图中的流图：

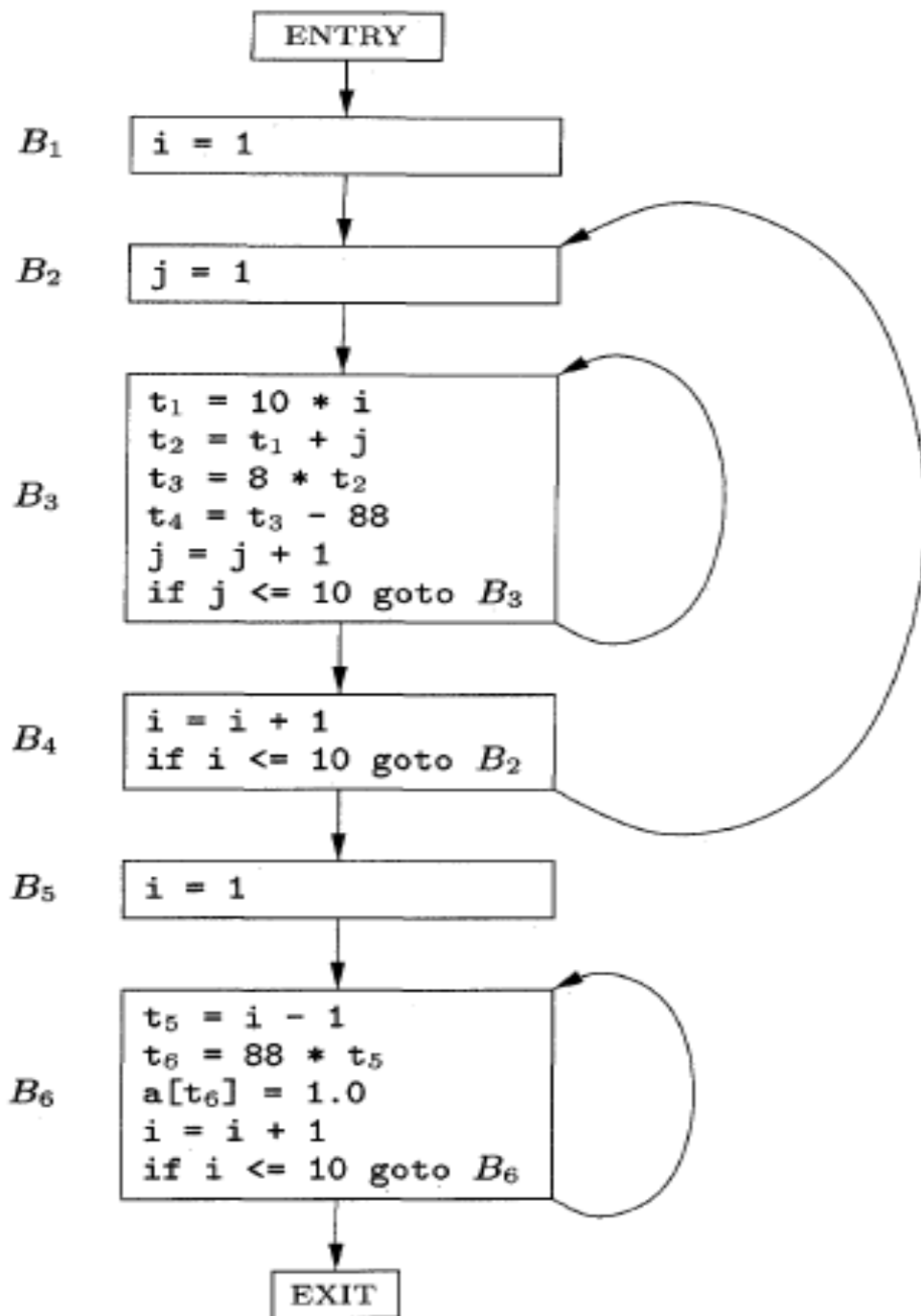
- (1) 计算支配关系
- (2) 构造支配节点树
- (3) 找出这个流图的自然循环



## 习题19.8

➤ 对下图中的流图：

- (1) 计算支配关系
- (2) 构造支配节点树
- (3) 找出这个流图的自然循环



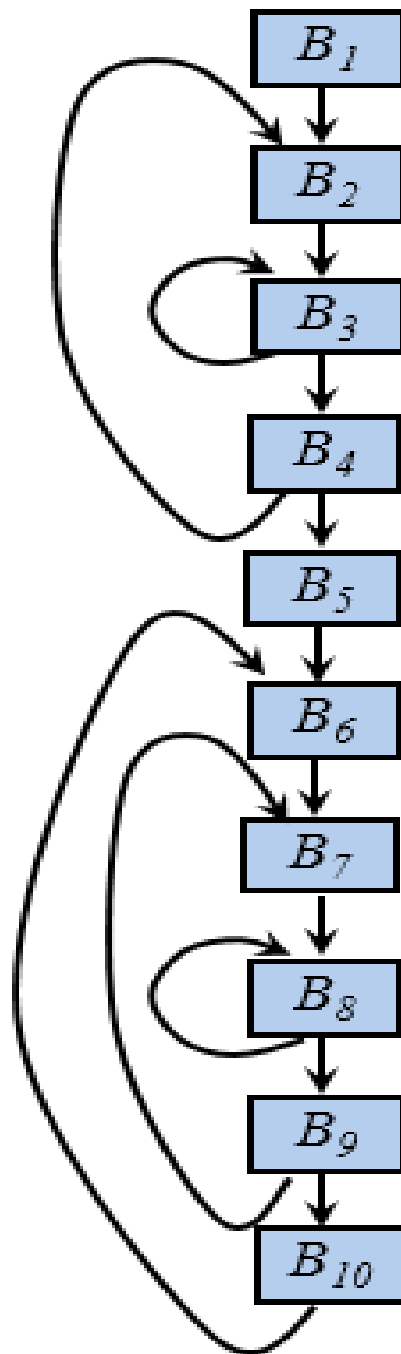
## 习题19.9

➤ 对下图中的流图：

➤ (1) 计算支配关系

➤ (2) 构造支配节点树

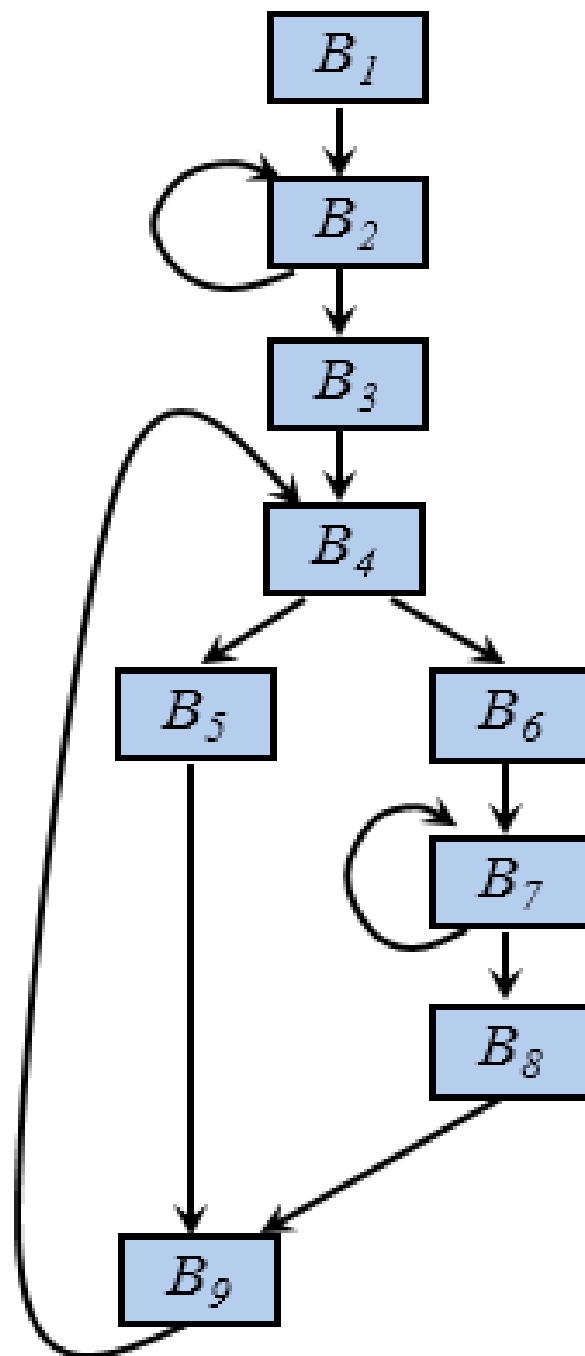
➤ (3) 找出这个流图的自然循环



## 习题19.10

➤ 对下图中的流图：

- (1) 计算支配关系
- (2) 构造支配节点树
- (3) 找出这个流图的自然循环





# 结束

---

哈尔滨工业大学 陈鄞

