第二章　一个微小编译器

1．　给定下面源程序，写出词法分析后的 TOKEN 表示：

```
begin     var X：real；
          var J：integer；
         read (J)；
          J：=J+(j*20)；
          X：=J-1；
          write(2*J+X)
end
```

(答案)

$begin	$var	($id, X)	$colon	$real	$
$line	$var	($id, J)	$colon	$int	$
$line	$read	$LParen	($id, J)	$RParen	$
$line	($id, J)	$assig	($id, J)	$plus	$L
($id, J)	$mult	($intC , 20)	$RParen	$semi	$
($id, X)	$assig	($id, J)	$minus	($intC , 1)	$
$line	$write	$LParen	($intC , 2)	$mult	($i
$plus	($id, X)	$RParen	$end	$eof	

(关闭)

2．　试写出上述程序的目标程序。

(答案)

```
INP   <J>
MUL    <J>   20   <t1>
ADD   <J>   <t1>  <t2>
STO   <t2>  <J>
SUB   <J>    1     <t3>
STO   <t3>  <X>
MUL    2    <J>    <t4>
ADD   <t4>  <X>   <t5>
OUT   <t5>
```

(关闭)

3. 出下面表达式的代码生成过程；
   (a) a*a+b*c+b
   (b) a*(a+b*c)+b

(答案)

语言栈	余下表达式	动作说明	目标代码
#	a*a+b*c+b#	Push(a)	
#a	*a+b*c+b#	Push(*)	
#a*	a+b*c+b#	Push(a)	
#a*a	+b*c+b#	产生代码;pop(3);push(t1)	MUL<a><a><t1>
#t1	+b*c+b#	Push(+)	
#t1+	b*c+b#	Push(b)	
#t1+b	*c+b#	Push(*)	
#t1+b*	c+b#	Push(c)	
#t1+b*c	+b#	产生代码;pop(3);push(t2)	MUL<b><c><t2>
#t1+t2	+b#	产生代码;pop(3);push(t3)	ADD<t1><t2><t3>
#t3	+b#	Push(+)	
#t3+	b#	Push(b)	

#t3+b	#	产生代码;pop(3);push(t4)	ADD<t3><b><t4>
#t4	#	结束	

语言栈	操作符栈	余下表达式	动作说明	目标代码
#	#	a*(a+b*c)+b#	PushOperand(a)	
#a	#	*(a+b*c)+b#	PushOperator(*)	
#a	#*	(a+b*c)+b#	PushOperator(Δ)	
#a	#*Δ	a+b*c)+b#	PushOperand(a)	
#aa	#*Δ	+b*c)+b#	PushOperator(+)	
#aa	#*Δ+	b*c)+b#	PushOperand(b)	
#aab	#*Δ+	*c)+b#	PushOperator(*)	
#aab	#*Δ+*	c)+b#	PushOperand(c)	
#aabc	#*Δ+*	)+b#	产生代码;PopOperand(2); PopOperator(1); PushOperand(t1);	MUL<b><c><t1
#aat1	#*Δ+	)+b#	产生代码;PopOperand(2); PopOperator(1); PushOperand(t2)	ADD<a><t1><t2
#at2	#*Δ	)+b#	PopOperator(1)	
#at2	#*	+b#	产生代码;PopOperand(2); PopOperator(1); PushOperand(t3)	MUL<a><t2><t3
#t3	#	+b#	PushOperator(+)	
#t3	#+	b#	PushOperand(b)	
#t3b	#+	#	产生代码;PopOperand(2); PopOperator(1); PushOperand(t4)	ADD<t3><b><t4
#t4	#	#	结束	

(关闭)

4. 考虑这样的目标代码生成算法：当常量为常数时不生成代码，而是由编译程序算出结果。例如，假如有语句
则生成代码：

```
MULT 10 b 1
STORE t a
```

提示：按原算法当要生成代码时，首先看分量是否都是常数，若不是，则完成原来的工作，否则
代码，而是计算值并把值压入语义信息栈，其他工作类似。

(答案)

算法：只需在目标代码生成子程序 ProceduceCode 中判断两个分量并进行相应的处理即可，

两个分量分别为 SemStack[top]和 SemStack[top-2]

运算符为 SemStack[top-1]

调用 IsConst()判断两个分量是否都是常数，若是调用 Compute()直接计算而不产生代码

```
ProduceCode:
      if  !Euqua(typ1,typ2)  then error;
      else {  if ( IsConst(Semstack[top]) && IsConst(SemStack[top-2]))
              then temp=Compute(SemStack[top].val,SemStack[top-2].val,SemStack[top-1
              else { temp= NewAddr; SendCode(OP,ADDR1,ADDR2,temp);}
              Pop(3);
              Push(temp,typ1);
           }
```

(关闭)

5. 设扩充条件语句 if〈表达式〉then〈语句〉else〈语句〉，其中表达式和语句定义同前。试说明编译程序应如

(答案)

(1) 对词法分析扩充

加入 TOKEN: $if $then $else

对 Scanner 加以扩充

第六行加入 ¡if¡=>GetToken($if)

¡then¡=>GetToken($then)

¡else¡=>G etToken($else)

(2) 对语法分析扩充

因为 then 或 else 后面是语句，所以形成语句嵌套，可以用一个子程序 Statement()处理单个

```
Procedure   Statement()
  Begin     ReadToken(token);
            case token  of
            ($if,_) =>{Match($if,18);Expr();Match($then,19);Statement();Match($semi,
                       Match($else,21);Statement();}
            ($write,_)=>同书
            ($read,_) =>同书
            ($id,_)   =>同书
             other    =>同书
  End
```

书中 Parse()子程序中，标号 LS 后(10-16 行)修改如下，其余不变

```
 LS:        BackToken();
            Statement();
```

(3) 对语义分析的扩充

在语法分析阶段已经对 if¡then¡else 的结构进行了分析,在语义分析阶段需要对 if 后的 <

判断是否为布尔表达式

(关闭)