编译第八周作业1

Pascal记录变量的ATG文法：

<svar decl> → <type> ↑t , i <entity> ↑n @svardef ↓ t, i, n@allocsv↓i ;

<type> ↑t, i → real ↑t, i |integer ↑t, i |character ↑t (<number>↑i)| logical ↑t, i ;

(其中，n为变量名，t为类型值，i为类型变量所需数据空间的大小)

输入文法：

<record del>→<entity>↑n = RECORD↑k@init↑m,s,a <comp\_list>↓a,m,s↑m',s' END

@insertsym↓n,k,m',s'

<comp\_list>↓a,m,s↑m',s'→<comp\_del>↓a↑L,a' @compinfo↓m,L↑m',s'

<comp\_dels>↓a',m',s'↑m'',s''

<comp\_dels>↓a,m,s↑m',s'→;<comp\_del>↓a↑L,a'@compinfo↓m,L↑m',s'<comp\_dels>| ε

<comp\_del>↓a↑L,a'→<name>↑c\_n := <type>↑t,L@insertcomp↓a,c\_n, t, L↑a'

(其中，n: 记录名 k：记录标识 m：分量个数 a:记录分量的符号表入口地址

s：各分量大小之和 l：分量大小 t：分量类型)

相应的语义动作程序：

procedure svardef( t, i, n );

j := tableinsert ( n, t, i ); //将有关信息填入符号表

if j == 0 //填表时要检查是否重名

then errmsg ( duplident , statementno);//输出相关错误信息

else if j == -1 //符号表已满

then errmsg( tblovflow, statementno);

abort(); //符号表溢出，编译失败，终止编译程序

end if;

end svardef;

procedure allocsv( i ); //存储分配

codeptr := codeptr + i ; //codeptr为分配地址指针

//设简单变量所要求的空间大小为i，因此可更新codeptr的内容

end allocsv**；**

procedure init; //初始化

m: = 0; /\* 分量计数器清0 \*/

s:= 0; /\* 分量大小计数器清0 \*/

a:= NewEntry; /\*可用表项的入口地址\*/

end;

procedure insertsym(n,k,m,s); //把记录类型名、记录标识、分量个数和记录所需空间大小填入符号表

procedure compinfo(m,L)

m:= m+1; /\*分量个数＋ 1\*/

s := s + L; /\* 分量大小被统计 \*/

return (m, s );

end

procedure insertcomp(a, c, \_n, t, L )

在a所指示的符号表项填入分量的名字、类型和大小

a = a ＋ 1;

return a; //将下一个可用表项的入口返回

end;

编译第八周作业2

写出for语句在执行循环体之前先做循环条件测试的属性翻译文法及其 处理动作程序。

ATG文法：

1.<for loop>→<for head>↑a, f, r < rest of loop>↓a, f, r

2.<for head>↑a, f, r →for <id>↑a := <expr>@initload

to <expr> @store↑e2@loadid↓a @genjmp↑s by @labgen↑r

@loadid↓e2 @loadid↓a<expr>@compare↓a,s↑f

3.<rest of loop>↓a, f, r →do <stat list> end for @retbranch↓r@labemit↓f

处理动作程序：

LDA, (<id>)

LOD, <expr1>

STN ---@ initload

LOD, <expr2>

STO, E2---@store

LOD, <id>--@loadid

JMP test---@genjmp↑s s:test

loop: ----labgen↑r r:loop

LOD, E2

LOD, <id>

LOD, <expr3>

ADD

STO, <id>

test: //compare略有改动，先设置s,再生成f-end\_loop,并生成BGT

BGT, end\_loop

<statement>

…

JMP, loop

end\_loop