**Val:char real integer boolean function procedure array const**

**idnumber:id的序号**

program → program\_1

program\_1→ id ( identifier\_list ) ; program\_body . program\_1 | empty

program\_body → const\_declarations type\_declarations var\_declarations

subprogram\_declarations compound\_statement

identifier\_list → id identifier\_list\_1

**id.in=identifier\_list.in//继承**

**identifier\_list\_1.in=identifier\_list.in**

**在符号表中增添id的表项**

**Identifier\_list.{idnumber1,idnumber2...}=id.idnumber+identifier\_list\_1.{idnumber1,...}**

**Id.val=identifier\_list\_1.val;**

identifier\_list\_1 → , id identifier\_list\_1 | empty

**id.in= identifier\_list\_1.in**

**identifier\_list\_1.in= identifier\_list\_1.in**

**在符号表中增添id的表项**

**Identifier\_list\_1.{idnumber1,idnumber2...}=id.idnumber+identifier\_list\_1.{idnumber1,...}**

const\_declarations → const const\_declaration | empty

const\_declaration → id = const\_variable ; const\_declaration\_1

**id.val=const//在符号表中添加id的表项**

**const\_variable.idnumber=id.idnumber;**

const\_declaration\_1 → ; id = const\_variable ; const\_declaration\_1 | empty

**id.val=const//在符号表中添加id的表项**

**const\_variable.idnumber=id.idnumber;**

const\_variable → + const\_variable\_1 | - const\_variable\_1 | const\_variable\_1 | ' letter '

**const\_variable.idnumber符号表中内容修改+-‘letter’**

**const\_variable\_1.idnumber=const\_variable.idnumber**

const\_variable\_1 → id | num

**根据idnumber修改符号表中内容**

type\_declarations → type type\_declaration | empty

type\_declaration → id = type ; type\_declaration\_1

**id.val=type.val;**

**在符号表中增添id的表项**

**Type.idnumber=id.idnumber;**

type\_declaration\_1 → ; id = type ; type\_declaration\_1 | empty

**id.val=type.val;**

**在符号表中增添id的表项**

**Type.idnumber=id.idnumber;**

type → standard\_type | record record\_body end | array [periods] of type

**type.val=standard\_type.val**

**type.val=record\_body.val**

**type.val=array**

**standard.idnumer=type.idnumber**

**record\_body.idnumber=type.idnumber**

**根据idnumber修改表中内容，同时type(最右).idnumber=type.idnumber**

**periods.idnumber=type.idnumber**

standard\_type → integer | real | Boolean | char

**stadard\_type.val=integer|real|boolean|char**

**根据idnumber修改符号表中内容。**

record\_body → var\_declaration | empty

**var\_declaration.idnumber=record\_body.idnumber;**

periods → period periods\_1

**继承idnumber属性**

periods\_1 → , period periods\_1 | empty

**继承idnumber属性，同时使维度+1**

period → + period\_1 | - period\_1 | period\_1 | ' letter ' .. const\_variable

period\_1 → id .. const\_variable | num .. const\_variable

//上面两个暂时不知道该咋处理

var\_declarations → var var\_declaration | empty

var\_declaration → identifier\_list : type ; var\_declaration\_1

**identifier\_list.val=type.val**

**type.idnumber{}=identifier\_list.idnumber{}**

var\_declaration\_1 → ; identifier\_list : type ; var\_declaration\_1 | empty

**identifier\_list.val=type.val**

**type.idnumber{}=identifier\_list.idnumber{}**

subprogram\_declarations → subprogram\_declarations\_1

subprogram\_declarations\_1 → subprogram\_declaration ; subprogram\_declarations\_1 | empty

subprogram\_declaration → function id formal\_parameter : standard\_type ; program\_body | procedure id formal\_parameter ; program\_body

**id.val=function**

**id.val=procedure**

**formal\_parameter\standard\_type\program\_body都获取idnumber和val的属性**

formal\_parameter → ( parameter\_lists ) | empty

**parameter\_lists继承idnumber和val**

parameter\_lists → parameter\_list parameter\_lists\_1

**继承idnumber和val**

parameter\_lists\_1 → ; parameter\_list parameter\_lists\_1 | empty

**继承idnumber 和val同时id在符号表中的自变量个数属性被修改。**

parameter\_list → var\_parameter | value\_parameter

**继承idnumber和val**

var\_parameter → var value\_parameter

**继承idnumber和val**

value\_parameter → identifier\_list : standard\_type

**继承idnumber，val**

compound\_statement → begin statement\_list end

statement\_list → statement statement\_list\_1

statement\_list\_1 → ; statement statement\_list\_1 | empty

statement → id statement\_1 | compound\_statement | if expression then statement else\_part | case expression of case\_body end | while expression do statement | repeat statement\_list until expression | for id assignop expression updown expression do statement | empty

**statement\_1.val=id.val;//根据id获取具体的idnumber**

**expression.val=boolean;**

statement\_1 → id\_varparts assignop expression | call\_procedure\_statement\_1

**id\_varparts.val=statements\_1.val**

**expression.val=statements\_1.val**

**判断statement.val是不是过程，不是的话报错使得话继承idnumber属性**

id\_varparts → id\_varparts\_1

**继承val**

id\_varparts\_1 → [ expression\_list ] id\_varparts\_1 | . id id\_varparts\_1 | empty

**不懂，唉 感觉不太对啊**

else\_part → else statement | empty

case\_body → branch\_list | empty

branch\_list → branch branch\_list\_1

branch\_list\_1 → ; branch branch\_list\_1 | empty

branch → const\_list : statement

const\_list → const\_variable const\_list\_1

const\_list\_1 → , const\_variable const\_list\_1 | empty

updown → to | downto

call\_procedure\_statement → id call\_procedure\_statement\_1//这一句没有用

call\_procedure\_statement\_1 → ( expression\_list ) | empty

**继承idnumber**

expression\_list → expression expression\_list\_1

**继承idnumber，同时计算参数数目，并把是第几号参数保存，超出就报错**

**标记为参数表达式**

expression\_list\_1 → , expression expression\_list\_1 | empty

**同上**

expression → simple\_expression expression\_1

**标记为参数表达式，继承是第几号参数属性**

**非参数的情况。**

expression\_1 → relop simple\_expression | empty

**参数表达式只可以推出为空，否则报错**

simple\_expression → term simple\_expression\_1 | + term simple\_expression\_1 | - term simple\_expression\_1

**标记为参数表达式，继承是第几号参数属性**

**Simple\_expression.val=term.val;**

**如果term和simple\_expression\_1不匹配，报错。**

simple\_expression\_1 → addop term simple\_expression\_1 | empty

**参数表达式只可以推出为空，否则报错**

**Simple\_expression\_1.val=term.val;**

**Term.val与simple\_expression\_1.val不匹配，报错。**

term → factor term\_1

**如果factor.val与term\_1.val不匹配，需要报错。**

**标记为参数表达式，继承是第几号参数属性**

**Term.val=factor.val**

term\_1 → mulop factor term\_1 | empty

**参数表达式只可以推空，否则报错**

**Term\_1.val=factor.val**

**如果factor.val与后面的term\_1不匹配，需要报错。**

factor → id factor\_1 | num | ' letter ' | ( expression ) | not factor

**标记为参数表达式。并根据第几号参数与符号表中进行匹配，出错报错。**

**factor.val=id.val/num.val/char/expresstion.val/boolean**

**factor(you).val=boolean;**

factor\_1 → id\_varparts | ( expression\_list ) | empty

unsign\_const\_variable → id | num | ' letter '

**unsign\_const\_variable=id.val/num.val/char//这一句也没有用**

addop → + | - | or

mulop → \* | / | div | mod | and

relop → = | <> | < | <= | > | >=

assignop → :=

**num.val=integer.**

**不支持array的复杂类型**

**不支持函数的嵌套调用**

**不支持……**