F ::= X

X ::= V ; X

X ::= C ; X

X ::= void main ( ) { L }

ID{

string name;

string type;

bool changable;

string value;

int level;

}

Cid{//常量

string type;

string value;

}

V{

string type;

}

T{

string type;

}

S{

vector<int> nextlist;

}

L{

vector<int> nextlist;

}

M{

int quad;

}

N{

vector<int> nextlist;

}

A{

}

E{

string place;

string type;

vector<int> truelist;

vector<int> falselist;

}

V ::= V1 , id {id.type = V1.type;V.type = V1.type;将id加入变量表}

V ::= var T id {id.type = T.type;V.type = T.type;将id加入变量表}

C ::= const T id = id1 {判断T和id1的type是否相同，若不同，报2错;

gen(id.value = id1.value);id.type = T.type;id.changable = false;

将id加入变量表}

C ::= const T id = cid {判断T和cid的type是否相同，若不同，报2错;

gen(id.value = cid.value);id.type = T.type;id.changable = false;

将id加入变量表}

T ::= int {T.type = int}

T ::= real {T.type =real}

T ::= bool

L ::= L1 M S {backpatch(L1.nextlist,M.quad);L.nextlist=S.nextlist;}

L ::= S {L.nextlist=S.nextlist;}

S ::= A ; {S.nextlist=null}

S ::= V ; {S.nextlist=null}

S ::= C ; {S.nextlist=null}

S ::= if ( E ) M1 { L1 } N else M2 { L2 } {backpatch(E.truelist,M1.quad);

backpatch(E.falselist,M2.quad);

S.nextlist = merge(L1.nextlist,L2.nextlist,N.nextlist);}

S ::= if ( E ) M { L } {backpatch(E.truelist,M.quad);

S.nextlist = merge(E.falselist,L.nextlist);}

S ::= while M1 ( E ) M2 { L } {backpatch(L.nextlist,M1.quad);

backpatch(E.truelist,M2.quad);

S.nextlist = E.falselist;

gen(j,\_,\_,M1.quad);}

M ::= {M.quad = nextquad;}

N ::= {N.nextlist = nextquad;gen(j,\_,\_,-1);}

// A ::= id = cid {判断id和cid类型是否相同，若不相同报2错;

gen(:=,cid.name,\_,id.name);}

A ::= id = E {判断id和E类型是否相同，若不相同报2错;

gen(:=,E.place,\_,id.name);}

A ::= id ++ {判断id是不是int或者real类型，若不是报2错； gen(+,id.place,1,id.place);}//这里是直接一步

A ::= id -- {判断id是不是int或者real类型，若不是报2错； gen(1,id.place,1,id.place);}

E ::= E1 + E2 {判断E1，E2是不是整形或实型，若不是，报2错；

E.place = newtemp;gen(+,E1.place,E2.place,E.place)

若E1，E2一个是int，一个是real，则E.type= real,

否则E.type=E1.type}

E ::= E - E

E ::= E \* E

E ::= E / E

E ::= E % E {这个其他的和相加相似，但要求右边两个E都是int}

E ::= ! E1 {判断E1是不是bool类型，不是则报2错;

E.place = newtemp;gen(!,E1.place,\_,E.place);E.type = bool;

E.truelist=E1.falselist;E.falselist=E1.truelist;}

E ::= - E1 {判断E1是不是int/real类型，不是则报2错;

E.place = newtemp;gen(@,E1.place,\_,E.place);E.type =E1.type;}

E ::= ( E1 ) {E.truelist = E1.truelist;E.falselist = E1.falselist;

代码上可以直接不删E1，让s=e1，删掉()继续做}

E ::= id {E.place = id.name;E.type=id.type;

若id不是bool;E.truelist=E.falselist=null;

若id是bool;E.truelist= nextquad;E.falselist=nextquad+1;}

E ::= cid {E.place = newTemp;E.type=cid.type;

若cid不是bool;E.truelist=E.falselist=null;

若cid是bool;E.truelist= nextquad;E.falselist=nextquad+1;}

E ::= E1 || M E2 {首先判断E1，E2是不是bool型，不是则报2错;

E.place=newtemp;

backpatch(E1.falselist,M.quad);

E.truelist=merge(E1.truelist,E2.truelist);

E.falselist=E2.falselist;

E.type=bool;

E ::= E1 && M E2 {首先判断E1，E2是不是bool型，不是则报2错;

E.place=newtemp;

backpatch(E1.truelist,M.quad);

E.falselist=merge(E1.falselist,E2.falselist);

E.truelist=E2.truelist;

E.type=bool;

E ::= E1 > E2 {E.place=newtemp;

E.type=bool;E.truelist= nextquad;E.falselist=nextquad+1;

gen(j>,E1.place,E2.place,-1);gen(j,\_,\_,-1);

}

E ::= E < E {E.place=newtemp;

E.type=bool;E.truelist= nextquad;E.falselist=nextquad+1;

gen(j<,E1.place,E2.place,-1);gen(j,\_,\_,-1);

}

E ::= E >= E {E.place=newtemp;

E.type=bool;E.truelist= nextquad;E.falselist=nextquad+1;

gen(j>=,E1.place,E2.place,-1);gen(j,\_,\_,-1);

}

E ::= E <= E {E.place=newtemp;

E.type=bool;E.truelist= nextquad;E.falselist=nextquad+1;

gen(j<=,E1.place,E2.place,-1);gen(j,\_,\_,-1);

}

E ::= E == E {E.place=newtemp;

E.type=bool;E.truelist= nextquad;E.falselist=nextquad+1;

gen(j==,E1.place,E2.place,-1);gen(j,\_,\_,-1);

}

E ::= E != E {E.place=newtemp;

E.type=bool;E.truelist= nextquad;E.falselist=nextquad+1;

gen(j!=,E1.place,E2.place,-1);gen(j,\_,\_,-1);

}