简单四元式的定义：

算数运算四元式（+，-，\*，/）

错误检查：

若操作数为标识符，若标识符首次出现，即未定义，则错误。

（op，num1,num2,des）

赋值语句四元式

（=，res,\_\_\_,des）

变量声明四元式：

无四元式，仅需填符号表。

函数声明四元式：

无四元式，仅需填符号表。

函数定义四元式：

错误检查：

在函数声明中寻找同名函数 检查参数列表即返回类型是否合理

（FunDefBegin,NameP,type,\_\_）

(FunReturn,argument,\_\_,\_\_)

(FunDefEnd,NameP,\_\_,\_\_)

包含语义动作的文法

(维护一个指向符号表的指针栈 栈顶始终指向待填项)

（维护type栈，栈顶保存立马要填写的类型）

（维护四元式栈，栈顶保存待填写四元式的指针）

（操作数栈）

program->ExtDefList   
ExtDefList->ExtDef ExtDefList | @  
ExtDef->Specifier FunDec CompSt  
Specifier->TYPE  
TYPE->”int” (记录返回值类型 type进符号栈) | “float” (同前)  
FunDec->ID（将id\_token，将指向符号表的指针入栈，生成函数四元式，取类型栈顶一个类型，填类型、种类：函数到符号表，填四元式）( VarList ) （填函数表的参数列表）| ID ()（同前）  
CompSt->{ DefList StmtList }（类型栈弹出栈顶元素）（生成函数定义结束四元式）  
DefList->Def DefList | @  
Def->Specifier DecList ; （类型栈弹出栈顶元素）  
DecList->Dec | Dec , DecList  
Dec->VarDec （操作符弹栈）| VarDec = Exp （生成赋值四元式）  
VarDec->ID（将type填入 符号表指定ID在符号表中 类型位 不出栈，进操作符栈，此时会将其从token栈中弹出）（错误检查，作用域）  
StmList->Stmt StmList|@  
Stmt->Exp ; | “return” Exp ;（生成函数返回四元式）

 Exp->ID(push)= Exp（错误检查，首次出现的标识符，报错） （生成赋值四元式）| Exp + Exp（生成赋值四元式） | Exp – Exp| Exp \* Exp |Exp / Exp | (Exp)   
|-Exp（生成算数四元式）| ID（push 操作数栈）（错误检查，首次出现的标识符，报错） | INT （同前 ）| FLOAT（同前 ）

新产生式：

消除左递归：

 Exp->ID(push)= Exp（生成赋值四元式） exp2 | (Exp) exp2 |-Exp （生成算数四元式）exp2| ID （push 操作数栈）（错误检查，首次出现的标识符，报错）exp2| INT（push 操作数栈） exp2| FLOAT（push 操作数栈） exp2

exp2->P Exp （）exp2|@

int main()

{

Int a,b;

A=2;

B=2\*5+a;

Return 0;

}