**<program>** -> <dec> <main> <dec>

**<main>** ->main ( ) { <mst> }

**<dec>** -> <struct\_dec> | <struct\_chidlren> | ε

**<struct\_dec>** -> <access\_modifier> struct ID <extends> { <struct\_body>}  
**<access\_modifier>** -> public | private

**<extends>** -> extends:ID | ε

**<struct\_body>** -> <struct\_chidlren> <struct\_body\_tail>

**<struct\_body\_tail>** -> <struct\_body> | ε  
**<struct\_chidlren>** -> DT ID <dt\_decORfunc\_dec> | <array\_dec> | <Constructor>

**<dt\_decORfunc\_dec>** -> <func\_dec> | <dt\_dec>  
**<func\_dec>** -> ( <param\_list> ) <Body>

**<param\_list>** -> <param> <param\_list\_tail> | ε

**<param\_list\_tail>** -> , <param> <param\_list\_tail> | ε

**<param>** -> DT ID

**<Body>** -> { <MST> }  
  
**<Constructor>** -> ID ( <param\_list> ) <Body>

**<dt\_dec> →** <var\_init> <var\_init\_tail> ; //int a; //

**<var\_init> →** = <Const\_or\_ID> | ε // int a =5;//

**<var\_init\_tail> →** ,ID <var\_init> <var\_init\_tail> | ε //int a=b;//

// int a ,b;//

**<Const\_or\_ID>** -> Const | ID //int a =5, b=5;//

// int a=b,c=d;//

**<array\_dec>** -> <arr\_type> ID [] = { <arr\_const\_or\_id> };

**<arr\_type>** -> DT | ID

**<arr\_const\_or\_id>** -> ε | <Const\_or\_ID> | ID , | Const ,

**<SST>** -> <while\_loop> | <for\_loop> | <if\_else> | <do\_while> | <exp> | <try> | <throw> | <return> | <continue> | <break> | <dt\_dec> | <func\_dec> | <func\_call> |

**<MST>** -> <SST><MST> | ε  
  
**<while\_loop>** -> while (<cond>)<loop\_body>  
**<cond>** -> <Const\_or\_ID> | <Const\_or\_ID> <ROP> <Const\_or\_ID> | <exp>   
**<ROP>** -> RO1 | RO2  
**<loop\_body>** -> ; | <SST> | {<MST>}

**<for\_loop>** -> for (<F1><F2>;<F3>) <loop\_body>  
**<F1>** -> <dt\_dec> | <assign\_st> | ;  
**<F2>** -> <cond> | ε

**<F3>** -> ExprList> | ε

**<ExprList>** -> <Expr> <ExprListTail>

<ExprListTail> -><Expr> <ExprListTail> | ε

**<inc\_dec>** -> ID increase\_decrease

**<assign\_st>** -> ID = <assign\_options> ;

**<assign\_options>** -> <Const\_or\_ID> | <exp>

**<if>** -> if (<cond>) <loop\_body> <else>  
**<else>** -> else <loop\_body> | null

**<do\_while>** -> do <loop\_body> while (<cond>);

**<this>** ->this ID ;| this <func\_call> ;

**<func\_call>**-> ID ( <arg\_list> )

**<arg\_list>** -> <arg> <arg\_list\_tail> | ε

**<arg\_list\_tail>** -> , <arg> <arg\_list\_tail> | ε

**<arg>** -> Const | ID

**<Expr>** -> <AssignExpr>

<AssignExpr> → <OrExpr> <AssignExpr'>

<AssignExpr'> → <CAssign> <AssignExpr> | ε # CAssign = ( =, +=, -=, \*=, /=, %=)

<OrExpr> → <AndExpr> <OrExpr'>

<OrExpr'> → OR <AndExpr> <OrExpr'> | ε

<AndExpr> → <EqualityExpr> <AndExpr'>

<AndExpr'> → AND <EqualityExpr> <AndExpr'> | ε

<EqualityExpr> → <RelationalExpr> <EqualityExpr'>

<EqualityExpr'> → RO2 <RelationalExpr> <EqualityExpr'> | ε # RO2 = (==, !=)

<RelationalExpr> → <AdditiveExpr> <RelationalExpr'>

<RelationalExpr'> → RO1 <AdditiveExpr> <RelationalExpr'> | ε # RO1 = (<, >, <=, >=)

<AdditiveExpr> → <MultiplicativeExpr> <AdditiveExpr'>

<AdditiveExpr'> → PM <MultiplicativeExpr> <AdditiveExpr'> | ε # PM = (+, -)

<MultiplicativeExpr> → <UnaryExpr> <MultiplicativeExpr'>

<MultiplicativeExpr'> → MDM <UnaryExpr> <MultiplicativeExpr'> | ε # MDM = (\*, /, %)

<UnaryExpr> → inc\_dec <Primary> | ! <UnaryExpr> | <Primary>

<Primary> → <PrimaryRef> | <Constant> | ( <Expr> ) | <Ref> | <FuncCall> | <ArrayCall>

**<return>** -> return <return\_options> ;

**<return\_options>** -> ID | const | <exp> |null

**<continue>** -> continue;

**<break>** -> break;

**<try>** ->try { <MST> } <catch\_list>

**<catch\_list>** ->catch ( ID ) { <MST> } <catch\_list\_tail>

**<catch\_list\_tail>** ->catch ( ID ) { <MST> } <catch\_list\_tail> | ε

**<throw>** -> throw <throw\_options>:

**<throw\_options>** -> ID | Const | new ID ( <param\_list> )

**<const>** -> int |float|char|string|bool