<program> -> <structures> <declarations> <functions> <main-func>

<structures> -> <structure> <structures> | e

<structure> -> *create* **id** { <declarations> };

<declarations> -> <*P*> <type> <id-list> ; <declarations > | <arrayinit> |e
<*P*> -> *constant* | *e*

<id-list> -> **id** <T> | e
<T>-> **,** <id-list> |e

<functions> -> *function* <returntype> <funcname> : <funcsignature> <Y> <functions> | e
<Y> -> <block> | <try-block>
<funcsignature> -> *(* <args> *)*

<returntype> -> <type> | void

<args> -> <type> **id** <T2**>** | void <T2>
<T2>**->** , <args> | e

<funcname> -> **id**

<main-func> -> *main* <block>

**BLOCK**

<block> -> {<statements>}
<try-block> -> *try* <block> *catch* <block> *finally* <block>

**STATEMENTS**
<statements>-> <statement> <statements> | e
<statement> -> <assignstat> | <declarations> | <returnstat> | <ifstat>| <iterativestat> | <instat>|
<outstat> | *end;* | *next;* | <functioncall>

<assignstat> -> <arraystmt> := <Exp>;

<functioncall> -> *call* **id**(<toSend>);
<toSend> -> <arraystmt> <S>| <literal> <S>
<S> ->, <toSend> | e

<return stat> -> *return* <to return>;
<to return> -> <arraystmt> |<literal>

<ifstat> -> *if*(<condExp>){ <statements> } <elsestat>
<elsestat> -> *else* { <statements>} | e

<iterativestat> -> *while* (<condExp>){ <statements> }

<instat> -> *input* >> **id**;
<outstat> -> *output* << **id**;

<arrayinit> -> <arraypart> **id** ;
<arraypart> -> *array*(<types>)[<arithmeticexp>] <Z>
<Z> -> [<arithmeticexp>] <Z> | e
<arraystmt> -> **id** <X> | # id <X>
<X> -> [<arithmeticexp>]<X> | .id | e

<id> -> TK_Identifier

<Exp> -> <ORexp> | <functioncall>

<condExp> -> <ORexp>

<ORexp> -> <ANDexp> <F>
<F> -> || <ANDexp> <F> | e

<ANDexp> -> <equalityexp><G>
<G> -> &&<equalityexp><G> | e

<equalityexp> -> <relationalexp><H>
<H> -> <equalOp> <relationalexp> | e

<equalOp> -> == | !=

<relationalexp> -> <arithmeticexp> <J>
<J> -> <relOp> <arithmeticexp> | e
<relOp> -> > | < | <= | >=

<arithmeticexp> -> <addexp>

<addexp> -> <mulexp> <B>
<B> -> +<mulexp><B> | -<mulexp><B> | e

<mulexp> -> <bitexp><C>
<C> -> *<mulexp><C> | /<mulexp><C> | %<mulexp><C> | e

<bitexp> -> <unaryexp><D>
<D> -> <bitOp><bitexp><D> | e
<bitOp> -> & | |

<unaryexp> -> <notexp> <K> //only post increment or decrement
<K> -> inc | dec | e

<notexp> -> <notOp><simple> | <simple>
<notOp> -> !

<simple>-> <literal> | <arraystmt> | (<Exp>)
<literal> -> <integerliteral> | <booleanliteral> | <charliteral>

<booleanliteral> -> true | false
<integerliteral> -> TK_Integer
<charliteral> -> TK_Character

<type> ->  int | char | boolean | (user defined data type’s id)