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

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

<structure> -> *create* **TK\_Identifier** { <declarations> };

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

*<P> -> constant | e*

<id-list> -> **TK\_Identifier** <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> **TK\_Identifier** <T2**>** | void <T2>

<T2>**->** , <args> | e

<funcname> -> **TK\_Identifier**

<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* **TK\_Identifier**(<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* >> **TK\_Identifier**;

<outstat> -> *output* << **TK\_Identifier**;

<arrayinit> -> <arraypart> **TK\_Identifier** ;

<arraypart> -> *array*(<types>)[<arithmeticexp>] <Z>

<Z> -> [<arithmeticexp>] <Z> | e

<arraystmt> -> **TK\_Identifier** <X> | # TK\_Identifier <X>

<X> -> [<arithmeticexp>]<X> | .TK\_Identifier | e

<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)