Original

```
<program> ::= CD24 <id> <globals> <funcs> <mainbody>

<globals> ::= <consts> <types> <arrays>
<consts> ::= constants <initlist> | ε
<initlist> ::= <init> | <init> , <initlist>
<init>   ::= <id> = <expr>

<types> ::= typedef <typelist> | ε
<typelist> ::= <type> <typelist> | <type>
<type>  ::= <structid> def <fields> end
<type>  ::= <typeid> def array [ <expr> ] of <structid> end
<fields> ::= <sdecl> | <sdecl> , <fields>

<arrays> ::= arraydef <arrdecls> | ε
<arrdecls> ::= <arrdecl> | <arrdecl> , <arrdecls>
<arrdecl> ::= <id> : <typeid>

<funcs> ::= <func> <funcs> | ε
<func>  ::= func  <id> ( <plist> ) : <rtype> <funcbody>
<rtype> ::= <stype> | void
<plist>  ::= <params> |  ε
<params> ::= <param> | <param> , <params>
<param> ::= <sdecl> | <arrdecl> | const <arrdecl>
<funcbody> ::= <locals> begin <stats> end
<locals> ::= <dlist> |  ε
<dlist>  ::= <decl> | <decl> , <dlist>
<decl>  ::= <sdecl> | <arrdecl>

<mainbody> ::= main <slist> begin <stats> end CD24 <id>
<slist>  ::= <sdecl> | <sdecl> , <slist>
<sdecl> ::= <id> : <stype>  | <id> : <structid>

<stype> ::= int | float | bool

<stats>  ::= <stat> ; <stats> | <strstat> <stats> | <stat>; | <strstat>
<strstat> ::= <forstat> | <ifstat> | <switchstat> | <dostat>
<stat>   ::= <repstat> | <asgnstat> | <iostat> | <callstat> | <returnstat>

<forstat> ::= for ( <asgnlist> ; <bool> ) <stats> end
<repstat> ::= repeat ( <asgnlist> ) <stats> until <bool>
<dostat>        ::= do <stats> while ( <bool> ) end
<asgnlist> ::=  <alist> | ε
<alist>  ::= <asgnstat> | <asgnstat> , <alist>

<ifstat> ::= if ( <bool> ) <stats> end
<ifstat> ::= if ( <bool> ) <stats> else <stats> end
<ifstat> ::= if ( <bool> ) <stats> elif (<bool>) <stats> end

<switchstat>  ::= switch ( <expr> ) begin <caselist> end
<caselist>     ::= case <expr> : <stats> break ; <caselist> | default : <stats>

<asgnstat> ::= <var> <asgnop> <bool>
<asgnop>       :: == | += | -= | *= | /=

<iostat> ::= input <vlist> | print  <prlist> | printline <prlist>

<callstat> ::= <id> ( <elist> )  |  <id> ( )

<returnstat> ::= return void |  return <expr>

<vlist>  ::= <var> , <vlist> | <var>
<var>  ::= <id> | <id>[<expr>]  | <id>[<expr>] . <id>

<elist>  ::=  <bool> , <elist> | <bool>
 <bool>  ::= not <bool> | <bool><logop> <rel> | <rel>
<rel>  ::= <expr> <relop><expr> | <expr>
<logop> ::= and | or | xor
<relop>  ::=  == | != | > | <= | < | >=

<expr>  ::= <expr> + <term> | <expr> - <term> | <term>
<term>  ::= <term> * <fact> | <term> / <fact> | <term> % <fact> | <fact>
<fact>   ::= <fact> ^ <exponent> | <exponent>
<exponent> ::= <var> | <intlit> | <reallit> | <fncall> | true | false
<exponent> ::= ( <bool> )

<fncall> ::= <id> ( <elist> )  |  <id> ( )

<prlist> ::= <printitem> , <prlist> | <printitem>
<printitem> ::= <expr> | <string>
```

```
<callstat> ::= <id> ( <elist> )  |  <id> ( )
<asgnstat> ::= <var> <asgnop> <bool>
<asgnop>       :: == | += | -= | *= | /=
<varTail> [<expr>] <varTailTail> | ε
<varTailTail> . <id> | ε
```

```
<bool>  ::= not <bool> | <bool> <logop> <rel> | <rel>

not <bool>

not <bool> <logop> <rel>

not a & not b || c

with this there's not way to do not a & not b & not c

<bool> ::= not <rel><boolTail> | <rel><boolTail>
<boolTail> ::= neps | <logop> <bool>
fixes that and you can actually implement it
```

Matchset

```
<program> ::= CD24 <id> <globals> <funcs> <mainbody>

<globals> ::= <consts> <types> <arrays>
<consts> ::= constants <initlist> | ε

<initlist> ::= <init> <initlistTail>
<initlistTail> ::= , <initlist> | NEPS

<init>   ::= <id> = <expr>

<types> ::= typedef <typelist> | ε
<typelist> ::= <type> <typelistTail>
<typelistTail> ::= <typelist> | ε

//make the symbol table
<type>  ::= <structid> def <fields> end
<type>  ::= <typeid> def array [ <expr> ] of <structid> end

<fields> ::= <sdecl><fieldsTail>
<fieldsTail> ::= , <fields> | ε

<arrays> ::= arraydef <arrdecls> | ε

<arrdecls> ::= <arrdecl> <arrdeclsTail>
<arrdeclsTail> ::= , <arrdecls> | ε

<arrdecl> ::= <id> : <arrdeclTail>
<arrdeclTail> ::= <typeid>

<funcs> ::= <func> <funcsTail> | ε
<funcsTail> ::= <funcs> | NEPS

<func>  ::= func  <id> ( <plist> ) : <rtype> <funcbody>

<rtype> ::= <stype> | void

<plist>  ::= <params> |  ε

<params> ::= <param> <paramsTail>
<paramsTail> ::= , <params> |  ε

<param> ::= <paramDecl> | const <arrdecl>
<paramDecl> ::= <initDecl> <paramDeclTail>
<paramDeclTail> ::= <sdeclTail> | <typeid>

<funcbody> ::= <locals> begin <stats> end

<locals> ::= <dlist> |  ε

<dlist>  ::= <decl><dlistTail>
<dlistTail> ::=  , <dlist> | ε

<decl>  ::= <initDecl> <declTail>
<declTail> ::= <sdeclTail> | <arrdeclTail>

<mainbody> ::= main <slist> begin <stats> end CD24 <id>

<slist>  ::= <sdecl><slistTail>
<slistTail> , <slist> |  ε

<sdecl> ::= <initDecl> <sdeclTail>
<sdeclTail> ::= <stype> | <structid>

<initDecl> ::= <id> :

<stype> ::= int | float | bool

<stats>  ::= <stat>; <statsTail> | <strstat> <statsTail>

<statsTail> ::= <stats> | ε
<strstat> ::= <forstat> | <ifstat> | <switchstat> | <dostat>
<stat>   ::= <repstat> | <id><asgnorcallstat> | <iostat> | <returnstat>

<asgnorcallstat> ::= <asgnstatTail> | ( <callstatTail>

<forstat> ::= for ( <asgnlist> ; <bool> ) <stats> end
<repstat> ::= repeat ( <asgnlist> ) <stats> until <bool>
<dostat>  ::= do <stats> while ( <bool> ) end
<asgnlist> ::=  <asgnstat> <asgnlistTail>| ε
<asgnlistTail> ::= , <asgnlist> | ε

<ifstat> ::= if ( <bool> ) <stats> <ifstatTail>
<ifstatTail> ::= end | else <stats> end | elif (<bool>) <stats> end

<switchstat>  ::= switch ( <expr> ) begin <caselist> end
<caselist>     ::= case <expr> : <stats> break ; <caselistTail> | default : <stats>
<caselistTail>::= <caselist> | nEPS

<asgnstat> ::= <var> <asgnop> <bool>
<asgnstatTail> ::= <varTail> <asgnop> <bool>
<asgnop>   ::=  = | += | -= | *= | /=

<iostat> ::= input <vlist> | print  <prlist> | printline <prlist>
<callstatTail> ::= <elist>) | )

<returnstat> ::= return <returnstatTail>
<returnstatTail> ::= void | <expr>

<vlist>  ::= <var> <vlistTail>
<vlistTail> ::= , <vlist> | ε

<var>  ::= <id> <varTail>
<varTail> [<expr>] <varTailTail> | ε
<varTailTail> . <id> | ε

<elist>  ::=  <bool> <elistTail>
<elistTail> ::= , <elist> | ε

<bool> ::= not <rel><boolTail> | <rel><boolTail>
<boolTail> ::= neps | <logop> <bool>

<rel>  ::= <expr> <relTail>
<relTail> ::= <relop><expr> | ε

<logop> ::= and | or | xor
<relop>  ::=  == | != | > | <= | < | >=

<expr>  ::= <term> <exprTail>
<exprTail> ::=  + <expr> | - <expr> | ε

<term>  ::= <fact> <termTail>
<termTail> ::= * <term> | / <term> |  % <term> | ε

<fact>   ::= <exponent><factTail>
<factTail> ::= ^ <fact> | ε

<exponent> ::= <varorfncall> | <intlit> | <reallit> | true | false | ( <bool> )

<varorfncall> ::= <id> <varorfncallTail>
<varorfncallTail> ::= (<fncallTail> | <varTail>

//unused<fncall> ::= <id> ( <elist> )  |  <id> ( )
<fncallTail> ::= <elist> ) | )

<prlist> ::= <printitem> <prlistTail>
<prlistTail> ::= , <prlist> | ε

<printitem> ::= <expr> | <string>
```

First

```
{"TCD24"},//program

{"nEPS", "TCONS", "TTYPD", "TARRD"},//globals
{"nEPS", "TCONS"},//constants

{"TIDEN"},//initlist
{"nEPS","TCOMA"},//initlistTail

{"TIDEN"},//init

{"nEPS", "TTYPD"},//types
{"symSTRUCTID", "symTYPEID"},//typelist
{"nEPS", "symSTRUCTID", "symTYPEID"},//typelisttail


{"symSTRUCTID", "symTYPEID"},//type

{"TIDEN"},//fields
{"nEPS", "TCOMA"},//fieldsTail

{"nEPS", "TARRD"},//arrays

{"TIDEN"},//arrdecls
{"nEPS", "TCOMA"},//arrdeclstail

{"TIDEN"},//arrdecl
{"symTYPEID"},//arrdeclTail

{"nEPS", "TFUNC"},//funcs
{"nEPS", "TFUNC"},//funcstail

{"TFUNC"},//func

{"TINTG", "TFLOT", "TBOOL", "TVOID"},//rtype

{"NEPS", "TIDEN", "TCNST"},//plist

{"TIDEN", "TCNST"},//params
{"nEPS", "TCOMA"},//paramsTail

{"TIDEN", "TCNST"},//param
{"TIDEN"},//paramDecl
{"TINTG", "TFLOT", "TBOOL", "symSTRUCTID", "symTYPEID"},//paramDeclTail

{"TIDEN", "TBEGN"},//funcbody

{"nEPS", "TIDEN"},//locals

{"TIDEN"},//dlist
{"nEPS", "TCOMA"},//dlisttail

{"TIDEN"},//decl
{"TINTG", "TFLOT", "TBOOL", "symSTRUCTID", "symTYPEID"},//decltail

{"TMAIN"},//mainbody

{"TIDEN"},//slist
{"nEPS", "TCOMA"},//slistTail

{"TIDEN"},//sdecl
{"TINTG", "TFLOT", "TBOOL", "symSTRUCTID"},//sdecltail

{"TIDEN"},//initdecl

{"TINTG", "TFLOT", "TBOOL"},//stype

{"TREPT", "TIDEN", "TINPT", "TPRNT", "TPRLN", "TRETN", "TTFOR", "TIFTH", "TSWTH", "TTTDO"},
  //stats there is a duplicate of stats in the code to make less nodes
{"nEPS", "TREPT", "TIDEN", "TINPT", "TPRNT", "TPRLN", "TRETN", "TTFOR", "TIFTH", "TSWTH", "TTTDO"},//statstail
{"TTFOR", "TIFTH", "TSWTH", "TTTDO"},//strstat
{"TREPT", "TIDEN", "TINPT", "TPRNT", "TPRLN", "TRETN"},//stat

{"nEPS", "TLBRK", "TEQUL", "TPLEQ", "TMNEQ", "TSTEQ", "TDVEQ", "TLPAR"},//asgnorcallstat

{"TTFOR"},//forstat
{"TREPT"},//repstat
{"TTTDO"},//dostat
{"nEPS", "TIDEN"},//asgnlist
{"nEPS", "TCOMA"},//asgnlisttail

{"TIFTH"},//ifstat
{"TTEND", "TELSE", "TELIF"},//ifstattail

{"TSWTH"},//switchstat
{"TCASE", "TDFLT"},//caselist
{"nEPS", "TCASE", "TDFLT"},//caselisttail

{"TIDEN"},//asgnstat
{"TLBRK", "TEQUL", "TPLEQ", "TMNEQ", "TSTEQ", "TDVEQ"},//asgnstattail used in asgnorcallstat in place of asgnstat
{"TEQUL", "TPLEQ", "TMNEQ", "TSTEQ", "TDVEQ"},//asgnop

{"TINPT", "TPRNT", "TPRLN"},//iostat
{"TTNOT", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR", "TRPAR"},//callstattail

{"TRETN"},//returnstat
{"TVOID", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//returnstattail

{"TIDEN"},//vlist
{"nEPS", "TCOMA"},//vlisttail

{"TIDEN"},//var
{"nEPS", "TLBRK"},//vartail
{"nEPS", "TDOTT"},//vartailtail

{"TTNOT", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//elist
{"nEPS", "TCOMA"},//elisttail

{"TTNOT", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//bool
{"nEPS", "TTAND", "TTTOR", "TTXOR"},//booltail

{"TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//rel
{"nEPS", "TEQEQ", "TTNEQ", "TGRTR", "TGEQL", "TLESS", "TLEQL"},//reltail

{"TTAND", "TTTOR", "TTXOR"},//logop
{"TEQEQ", "TTNEQ", "TGRTR", "TGEQL", "TLESS", "TLEQL"},//relop

{"TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//expr
{"nEPS", "TPLUS", "TMINS"},//exprTail

{"TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//term
{"nEPS", "TSTAR", "TDIVID", "TPERC"},//termtail

{"TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//fact
{"nEPS", "TCART"},//facttail

{"TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//exponent

{"TIDEN"},//varorfncall
{"nEPS", "TLPAR", "TLBRK"},//varorfncalltail


{"TRPAR", "TTNOT", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//fncalltail

{"TSTRG", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"},//prlist
{"nEPS", "TCOMA"},//prlisttail

{"TSTRG", "TIDEN", "TILIT", "TFLIT", "TTRUE", "TFALS", "TLPAR"}//printitem
```