120101004 - Abhishek Goyal
120101021 - Dheeraj Khatri
120101046 - Ojas Deshpande

```
-----------------------------------------------------------------
                          GRAMMAR
-----------------------------------------------------------------
START -> OUTER MAINF OUTER

OUTER -> e |
 COMMENT OUTER |
 VARDEF OUTER |
 STRUCT1 OUTER |
FUNCTION OUTER


VOID -> void

INT -> int

MAINARG -> e |
int argc, char* argv[]

MAINF -> VOID MAIN(MAINARG){INNER} |
INT MAIN(MAINARG){INNER}

TYPE -> INT | BOOL | VOID | FLOAT | DOUBLE | STRUCT IDENTIFIER | TYPE* | CHAR


STRUCT1 -> STRUCT INDENTIFIER{MVARDEF VARDEF;};

MVARDEF -> MVARDEF VARDEF;|e


FUNCTION -> TYPE VARNAME(FARG){INNER}



INNER -> COMMENT INNER |
 LOOP INNER |
 CONDITIONAL INNER |
VARDEF INNER |
STRUCT INNER |
FCALL INNER |
RETURN INNER |
;|
e |
 INPUT INNER |
 OUTPUT INNER |
MATH INNER |
ASSIGN INNER
```

```
VARNAME -> *VARNAME | &VARNAME | IDENTIFIER BRACKET
BRACKET -> BRACKET[INTMATH] | e

MARG -> MARG TYPE VARNAME, | e
FARG -> MARG TYPE VARNAME

VARDEF ->TYPE MVAR VARNAME; |
 TYPE MVAR VARNAME = CONST; |
TYPE MVAR VARNAME = FCALL; |
TYPE MVAR VARNAME = RMATH; |
TYPE MVAR VARNAME = VARNAME;|
TYPE MVAR VARNAME = {MCONST CONST};


MVAR -> MVAR VARNAME, |
MVAR VARNAME = CONST, |
MVAR VARNAME = FCALL, |
MVAR VARNAME = RMATH, |
MVAR VARNAME = VARNAME, |
MVAR VARNAME = MCONST, |
e


MCONST -> MCONST CONST,|e


MATH -> VARNAME = RMATH; |
;|
TYPE VARNAME = RMATH; |
VARNAME OPERATOR= RMATH; |
VARNAME++; |
VARNAME--; |
++VARNAME; |
--VARNAME;


RMATH = VARNAME|
FCALL|
VARNAME++ |
VARNAME-- |
++VARNAME |
--VARNAME|
(RMATH)|
!RMATH |
CONST|
RADDSUB

RADDSUB -> RMULTDIV |
RADDSUB - RMULTDIV |
RADDSUB + RMULTDIV

RMULTDIV -> RMATH |
RMULTIDIV * RMATH |
RMULTIDIV / RMATH |
RMULTIDIV % RMATH


//for array indexing
INTMATH -> VARNAME|
FCALL|
(INTMATH) |
!INTMATH |
INTCONST|
ADDSUB|
VARNAME++ |
VARNAME-- |
++VARNAME |
--VARNAME|

ADDSUB -> MULTDIV |
ADDSUB - MULTDIV |
ADDSUB + MULTDIV

MULTDIV -> INTMATH |
MULTIDIV * INTMATH |
MULTIDIV / INTMATH |
MULTIDIV % INTMATH



CONDITIONAL -> IFN | SWITCHN


IFN -> IF(COND){INNER}ELSE


RELATIONALOPERATOR -> >|
<|
==|
!=|
<=|
>=



COND -> (COND)|
COND&&COND |
COND||COND |
!COND |
RMATH |
COND RELATIONALOPERATOR COND

ELSE -> ELIF (COND) {INNER} ELSE | ELSE {INNER} | e
```

```
SWITCHN -> SWITCH(COND){SWITCHINNER}

SWITCHINNER -> case CONST:{INNER} SWITCHINNER |
case CONST: INNER SWITCHINNER |
default:{INNER} WODEFAULT|
default: INNER WODEFAULT|
e


WODEFAULT -> CASE CONST:{INNER} WODEFAULT |
CASE CONST: INNER WODEFAULT|
e

ARGT -> MARGT RMATH
MARGT -> MARGT RMATH,| e
FCALL -> VARNAME(ARGT)



LOOP -> FORN | WHILEN | DOWHILEN

FL1-> MATH | e
FL2-> COND | e
FL3 -> VARNAME = RMATH |
TYPE VARNAME = RMATH|
VARNAME OPERATOR= RMATH |
VARNAME++|
VARNAME--|
++VARNAME|
--VARNAME


OPERATOR -> +|
-|
*|
/|
&|
^|
||
%

FORN -> FOR(FL1 FL2; FL3){INNER}

WHILEN->WHILE(COND){INNER}

DOWHILEN -> DO{INNER}WHILE(COND);

MIN -> MIN >>VARNAME |e


MOUT -> MOUT <<VARNAME |
MOUT << CONST |
```

MOUT << WHITESPACE |
e

INPUT -> IN MIN >> VARNAME;

OUTPUT -> OUT MOUT << VARNAME;|
OUT MOUT << CONST ;|
OUT MOUT << WHITESPACE;

CONST-> TRUE |
FALSE |
INTCONST|
FLOAT|
DOUBLE

LEX CODE

```
%{
#include <stdio.h>
%}



%%
[ \t]+ ;
main            {printf("MAIN ");}
if                      {printf("IF");}
elif            {printf("ELIF");}
else            {printf("ELSE");}
switch          {printf("SWITCH");}
case            {printf("CASE ");}
default         {printf("DEFAULT");}
do                      {printf("DO");}
return          {printf("RETURN ");}
void            {printf("VOID ");}
struct          {printf("STRUCT ");}
int             {printf("INT ");}
float           {printf("FLOAT ");}
bool            {printf("BOOL ");}
double                  {printf("DOUBLE ");}
char            {printf("CHAR ");}
for             {printf("FOR");}
while           {printf("WHILE");}
in                      {printf("IN");}
out                     {printf("OUT");}
\+                      {printf("+");}
```

```
\++                     {printf("++");}
\--                             {printf("--");}
\-                              {printf("-");}
\*                              {printf("*");}
\/                              {printf("/");}
\%                              {printf("%%");}
\=                              {printf("=");}
\==                             {printf("==");}
\<=                             {printf("<=");}
\>=                             {printf(">=");}
\<                              {printf("<");}
\>                              {printf(">");}
\!                              {printf("!");}
\&&                     {printf("&&");}
\^                              {printf("^");}
\|\|                    {printf("||");}
\<\<                    {printf("<<");}
\>\>                    {printf(">>");}

[0-9]+ |
[0-9]+\.[0-9]+ |
\.[0-9]+            {printf("CONSTANT");}
\"(\\.|[^\\"])*\" {printf("STRING");}
\( {printf("(");}
\) {printf(")");}
\{ {printf("{");}
\} {printf("}");}
\; {printf(";");}
\, {printf(",");}
\: {printf(":");}
[a-zA-Z][a-zA-Z0-9]* {printf("IDENTIFIER");}
[\n]            ;
(\/\*([^*]|[\r\n]|(\*+([^*/]|[\r\n])))*\*+\/)|(\/\/.*) {printf("COMMENT");}
%%



int yywrap(void){
        return 1;
}

int main(){
        yylex();
        return 0;
}
```