%{

/\*

\* =================================================================

\*

\* check2.l

\*

\* Programmer --- Patrick Fischer

\*

\*

\* =================================================================

\*/

#define DEBUG 0

#include "y.tab.h"

#include <stdio.h>

%}

%%

[\t ]+ { if(DEBUG) printf("WHITESPACE "); }

(m|M)(a|A)(i|I)(n|N) { if(DEBUG) printf("RWMAIN: %s\n", yytext);

return RWMAIN;

}

(e|E)(x|X)(i|I)(t|T) { if(DEBUG) printf("RWEXIT: %s\n", yytext);

return RWEXIT;

}

(i|I)(f|F) { if(DEBUG) printf("RWIF: %s\n", yytext);

return RWIF;

}

(e|E)(l|L)(s|S)(e|E) { if(DEBUG) printf("RWELSE: %s\n", yytext);

return RWELSE;

}

(w|W)(h|H)(i|I)(l|L)(e|E) { if(DEBUG) printf("RWWHILE: %s\n", yytext);

return RWWHILE;

}

(c|C)(o|O)(u|U)(n|N)(t|T)(i|I)(n|N)(g|G) { if(DEBUG) printf("RWCOUNTING: %s\n", yytext);

return RWCOUNTING;

}

(e|E)(n|N)(d|D) { if(DEBUG) printf("RWEND: %s\n", yytext);

return RWEND;

}

(d|D)(a|A)(t|T)(a|A) { if(DEBUG) printf("RWDATA: %s\n", yytext);

return RWDATA;

}

(a|A)(l|L)(g|G)(o|O)(r|R)(i|I)(t|T)(h|H)(m|M) { if(DEBUG) printf("RWALG: %s\n", yytext);

return RWALG;

}

[-+]?[0-9]+ { if(DEBUG) printf("LITINT: %s\n", yytext);

yylval.ival = atoi(yytext);

return LITINT;

}

[-+]?([0-9]\*\.?[0-9]+|[0-9]+\.)((e|E)[-+]?[0-9]+)? { if(DEBUG) printf("LITREAL: %s\n", yytext);

return LITREAL;

}

(i|I)(n|N)(t|T)(e|E)(g|G)(e|E)(r|R) { if(DEBUG) printf("RWINT: %s\n", yytext);

return RWINT;

}

(r|R)(e|E)(a|A)(l|L) { if(DEBUG) printf("RWREAL: %s\n", yytext);

return RWREAL;

}

(u|U)(p|P)(w|W)(a|A)(r|R)(d|D) { if(DEBUG) printf("RWUPWARD: %s\n", yytext);

return RWUPWARD;

}

(d|D)(o|O)(w|W)(n|N)(w|W)(a|A)(r|R)(d|D) { if(DEBUG) printf("RWDOWNWARD: %s\n", yytext);

return RWDOWNWARD;

}

(t|T)(o|O) { if(DEBUG) printf("RWTO: %s\n", yytext);

return RWTO;

}

(r|R)(e|E)(a|A)(d|D) { if(DEBUG) printf("RWREAD: %s\n", yytext);

return RWREAD;

}

(p|P)(r|R)(i|I)(n|N)(t|T) { if(DEBUG) printf("RWPRINT: %s\n", yytext);

return RWPRINT;

}

\"([^\\\"]|\\.)\*\" { if(DEBUG) printf("CHARSTRING: %s\n", yytext);

return CHARSTRING;

}

#.\*[\n]\* { if(DEBUG) printf("COMMENT: %s\n", yytext); }

! { if(DEBUG) printf("CARRETURN: %s\n", yytext);

return CARRETURN;

}

":=" { if(DEBUG) printf("ASSIGNOP: %s\n", yytext);

return ASSIGNOP;

}

, { if(DEBUG) printf("COMMA: %s\n", yytext);

return COMMA;

}

";" { if(DEBUG) printf("SEMICOLON: %s\n", yytext);

return SEMICOLON;

}

":" { if(DEBUG) printf("COLON: %s\n", yytext);

return COLON;

}

"(" { if(DEBUG) printf("LPAREN: %s\n", yytext);

return LPAREN;

}

")" { if(DEBUG) printf("RPAREN: %s\n", yytext);

return RPAREN;

}

"[" { if(DEBUG) printf("LBRACK: %s\n", yytext);

return LBRACK;

}

"]" { if(DEBUG) printf("RBRACK: %s\n", yytext);

return RBRACK;

}

"+" { if(DEBUG) printf("ADD: %s\n", yytext);

return ADD;

}

"-" { if(DEBUG) printf("MINUS: %s\n", yytext);

return MINUS;

}

"\*" { if(DEBUG) printf("MULT: %s\n", yytext);

return MULT;

}

"/" { if(DEBUG) printf("DIV: %s\n", yytext);

return DIV;

}

"<" { if(DEBUG) printf("LESS: %s\n", yytext);

return LESS;

}

">" { if(DEBUG) printf("GREATER: %s\n", yytext);

return GREATER;

}

"<=" { if(DEBUG) printf("LESSEQU: %s\n", yytext);

return LESSEQU;

}

">=" { if(DEBUG) printf("GREATEQU: %s\n", yytext);

return GREATEQU;

}

"=" { if(DEBUG) printf("EQUAL: %s\n", yytext);

return EQUAL;

}

"<>" { if(DEBUG) printf("NOTEQUAL: %s\n", yytext);

return NOTEQUAL;

}

"&" { if(DEBUG) printf("AND: %s\n", yytext);

return AND;

}

"|" { if(DEBUG) printf("OR: %s\n", yytext);

return OR;

}

"~" { if(DEBUG) printf("NOT: %s\n", yytext);

return NOT;

}

([a-zA-Z][a-zA-Z0-9]\*)+ { if(DEBUG) printf("VAR: %s\n", yytext);

yylval.sval = strdup(yytext); // Had to use strdup to not copy large chunks of text

//printf("yylval.sval = %s\n", yylval.sval);

return VAR;

free(yylval.sval); // Free up memory used by strdup

}

[\n]\* { if(DEBUG) printf("NEWLINE\n\n");

//return NEWLINE;

}

. { if(DEBUG) printf("Possible error: %s\n", yytext); }

%%