%{

#include <ctype.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char lexema[60];

void yyerror(char \*msg);

typedef struct {

  char nombre[60];

  double valor;

  int token;

  } tipoTS;

tipoTS TablaSim[100];

int nSim = 0;

typedef struct {

  int op;

  int a1;

  int a2;

  int a3;

} tipoCodigo;

int cx = -1;

tipoCodigo TCodigo[100];

void generaCodigo(int, int, int, int);

int localizaSimb(char \*, int);

void imprimeTablaSim();

void imprimeTablaCod();

int nVarTemp = 0;

int GenVarTemp();

int yylex();

int EsPalabraReservada(char[], int);

%}

%token PROGRAMA ID INICIO FIN NUM VARIABLE ASIGNAR SUMAR RESTAR MULTIPLICAR DIVIDIR PARENTESIS

%%

S: PROGRAMA ID ';' listadeclaracion INICIO listaInstr FIN '.';

listadeclaracion:  ;

listaInstr: instr listaInstr

          |  ;

instr: ID {$$ = localizaSimb(lexema, ID);} ':' '=' expr {generaCodigo(ASIGNAR, $2, $5, '-');} ';';

expr: expr '+' term {int i = GenVarTemp(); generaCodigo(SUMAR, i, $1, $3); $$ = i;}

    | expr '-' term {int i = GenVarTemp(); generaCodigo(RESTAR, i, $1, $3); $$ = i;}

    | term;

term: term '\*' term2 {int i = GenVarTemp(); generaCodigo(MULTIPLICAR, i, $1, $3); $$ = i;}

    | term '/' term2 {int i = GenVarTemp(); generaCodigo(DIVIDIR, i, $1, $3); $$ = i;}

    | term2;

term2: '(' expr ')' {int i = GenVarTemp(); generaCodigo(PARENTESIS, i, $2, '-'); $$ = i;};

     | NUM {$$ = localizaSimb(lexema, NUM);}

     | ID {$$ = localizaSimb(lexema, ID);};

%%

int GenVarTemp(){

  char t[60];

  sprintf(t, "\_T%d", nVarTemp++);

  return localizaSimb(t, ID);

}

void generaCodigo(int op, int a1, int a2, int a3){

  cx++;

  TCodigo[cx].op = op;

  TCodigo[cx].a1 = a1;

  TCodigo[cx].a2 = a2;

  TCodigo[cx].a3 = a3;

}

int localizaSimb(char \*nom, int tok) {

  int i;

  for(i = 0; i < nSim; i++) {

    if(!strcasecmp(TablaSim[i].nombre, nom))

      return i;

  }

  strcpy(TablaSim[nSim].nombre, nom);

  TablaSim[nSim].token = tok;

  if(tok == ID) TablaSim[nSim].valor = 0.0;

  if(tok == NUM) sscanf(nom, "%lf", &TablaSim[nSim].valor);

  nSim++;

  return nSim - 1;

}

void imprimeTablaSim(){

  int i;

  for(i = 0; i < nSim; i++){

    printf("%d  nombre=%s tok=%d valor=%lf\n", i, TablaSim[i].nombre, TablaSim[i].token, TablaSim[i].valor);

  }

}

void imprimeTablaCod(){

  int i;

  for(i = 0; i <= cx; i++){

    printf("%d  a1=%d a2=%d a3=%d\n", TCodigo[i].op, TCodigo[i].a1, TCodigo[i].a2, TCodigo[i].a3);

  }

}

void yyerror(char \*msg){

  printf("ERROR:%s\n",msg);

}

int EsPalabraReservada(char lexema[], int default\_token) {

  //strcmp considera may y minusc

  //strcasecmp ignora may de min

  if(strcasecmp(lexema,"Program")==0) return PROGRAMA;

  if(strcasecmp(lexema,"Begin")==0) return INICIO;

  if(strcasecmp(lexema,"End")==0) return FIN;

  if(strcasecmp(lexema,"Var")==0) return VARIABLE;

  return default\_token;

}

int yylex(){

  char c;

  int i;

  while(1) {

    c = getchar();

    if(c == ' ') continue;

    if(c == '\t') continue;

    if(c == '\n') continue;

    if(isdigit(c)) {

      i = 0;

      do{

        lexema[i++] = c;

        c = getchar();

      } while(isdigit(c));

      ungetc(c, stdin);

      lexema[i] = '\0';

      return NUM;

    }

    if(isalpha(c)){

      i = 0;

      do{

        lexema[i++] = c;

        c = getchar();

      } while(isalnum(c));

     ungetc(c, stdin);

     lexema[i] = '\0';

     return EsPalabraReservada(lexema, ID);

    }

    return c;

  }

}

int main(){

  if(!yyparse()) printf("La cadena es valida\n");

  else printf("La cadena es invalida\n");

  printf("tabla de simbolos\n");

  imprimeTablaSim();

  printf("tabla de codigos\n");

  imprimeTablaCod();

  return 0;

}



