%{

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <math.h>

char lexema[60];

void yyerror(char \*msg);

int yylex();

typedef struct {

  char nombre[60];

  double valor;

  char strval[60];

  int token;

  int tokenaux;

} 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();

int nVarTemp = 0;

int GenVarTemp();

void InterpretaCodigo();

int errorLine = 0;

%}

%token PROGRAMA INICIO FIN INICIOALGORITMO FINALGORITMO NUM ID CADENA VAR

%token SI SINO ENTONCES MIENTRAS HACER CALCULAR DESDE AUMENTADO EN HASTA SALIDA

%token ASIGNAR SUMAR RESTAR MULTIPLICAR DIVIDIR PARENTESIS POTENCIA UMENOS

%token SALTAR1 SALTAR2 SALTAR\_FOR BUCLE\_WHILE BUCLE\_FOR IMPRIMIR

%token MAYOR MENOR IGUAL DESIGUAL MAYOR\_IGUAL MENOR\_IGUAL

%%

programa: PROGRAMA ID '{' listaInstruccion '}';

listaInstruccion: instruccion

                | listaInstruccion instruccion;

instruccion: VAR list\_asig ';';

list\_asig: list\_asig ',' asig

         | asig;

asig: ID {$$=localizaSimb(lexema,ID);} '=' expr {generaCodigo(ASIGNAR,$2,$4,'-');};

instruccion: SI cond {generaCodigo(SALTAR1,$2,'?','-');$$=cx;} '[' listaInstruccion ']' {generaCodigo(SALTAR2,$2,'?','-');$$=cx;} {TCodigo[$3].a2=cx+1;} otro {TCodigo[$7].a2=cx+1;};

otro: SINO '[' listaInstruccion ']'

    |  ;

// instruccion: MIENTRAS cond {generaCodigo(SALTAR1,$2,'?','-');$$=cx;} HACER INICIO bloque {generaCodigo(BUCLE\_WHILE,$3,'-','-');}{TCodigo[$3].a2=cx+1;} FIN;

instruccion: CALCULAR DESDE ID {$$=localizaSimb(lexema,ID);} AUMENTADO EN ID {$$=localizaSimb(lexema,ID);} HASTA ID {$$=localizaSimb(lexema,ID);} {generaCodigo(SALTAR\_FOR,$4,$11,'?');$$=cx;} '[' listaInstruccion ']' {generaCodigo(BUCLE\_FOR,$4,$7,$12);TCodigo[$12].a3=cx+1;};

// instruccion: PARA ID {$$=localizaSimb(lexema,ID);} HASTA expr {generaCodigo(SALTAR\_FOR,$3,$5,'?');$$=cx;} INICIO bloque {generaCodigo(BUCLE\_FOR,$3,$6,'-');}{TCodigo[$6].a3=cx+1;} FIN;

// bloque: listaInstruccion

//       | instruccion;

instruccion: asig ';';

instruccion: SALIDA expr {generaCodigo(IMPRIMIR,$2,'-','-');};

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

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

expr: term;

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

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

term: term2;

term2: '-' term3 {int i = GenVarTemp(); generaCodigo(UMENOS,i,$2,'-'); $$=i;}

     | term3 '\*' '\*' term2 {int i = GenVarTemp(); generaCodigo(POTENCIA,i,$1,$4); $$=i;}

     | term3;

term3: '(' expr ')'{int i = GenVarTemp(); generaCodigo(PARENTESIS,i,$2,$2); $$=i;}

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

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

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

cond: expr '=' '=' expr {int i = GenVarTemp(); generaCodigo(IGUAL,i,$1,$4); $$=i;};

cond: expr '<' '>' expr {int i = GenVarTemp(); generaCodigo(DESIGUAL,i,$1,$4); $$=i;};

cond: cond2;

cond2: expr '<' expr {int i = GenVarTemp(); generaCodigo(MENOR,i,$1,$3); $$=i;};

cond2: expr '<' '=' expr {int i = GenVarTemp(); generaCodigo(MENOR\_IGUAL,i,$1,$4); $$=i;};

cond2: expr '>' expr {int i = GenVarTemp(); generaCodigo(MAYOR,i,$1,$3); $$=i;};

cond2: expr '>' '=' expr {int i = GenVarTemp(); generaCodigo(MAYOR\_IGUAL,i,$1,$4); $$=i;};

%%

void InterpretaCodigo(){

  int i,a1,a2,a3,op,j,temp;

  for(i=0; i<=cx; i++){

    op = TCodigo[i].op;

    a1 = TCodigo[i].a1;

    a2 = TCodigo[i].a2;

    a3 = TCodigo[i].a3;

    if(op==MAYOR){

      TablaSim[a1].tokenaux=MAYOR;

      if(TablaSim[a2].valor>TablaSim[a3].valor) TablaSim[a1].valor=1;

      else TablaSim[a1].valor=0;

    }

    if(op==MENOR){

      TablaSim[a1].tokenaux=MENOR;

      if(TablaSim[a2].valor<TablaSim[a3].valor) TablaSim[a1].valor=1;

      else TablaSim[a1].valor=0;

    }

    if(op==MAYOR\_IGUAL){

      TablaSim[a1].tokenaux=MAYOR\_IGUAL;

      if(TablaSim[a2].valor>=TablaSim[a3].valor) TablaSim[a1].valor=1;

      else TablaSim[a1].valor=0;

    }

    if(op==MENOR\_IGUAL){

      TablaSim[a1].tokenaux=MENOR\_IGUAL;

      if(TablaSim[a2].valor<=TablaSim[a3].valor) TablaSim[a1].valor=1;

      else TablaSim[a1].valor=0;

    }

    if(op==IGUAL){

      TablaSim[a1].tokenaux=IGUAL;

      if(TablaSim[a2].valor==TablaSim[a3].valor) TablaSim[a1].valor=1;

      else TablaSim[a1].valor=0;

    }

    if(op==DESIGUAL){

      TablaSim[a1].tokenaux=DESIGUAL;

      if(TablaSim[a2].valor!=TablaSim[a3].valor) TablaSim[a1].valor=1;

      else TablaSim[a1].valor=0;

    }

    if(op==SUMAR) TablaSim[a1].valor = TablaSim[a2].valor + TablaSim[a3].valor;

    if(op==RESTAR) TablaSim[a1].valor = TablaSim[a2].valor - TablaSim[a3].valor;

    if(op==MULTIPLICAR) TablaSim[a1].valor = TablaSim[a2].valor \* TablaSim[a3].valor;

    if(op==DIVIDIR) TablaSim[a1].valor = TablaSim[a2].valor / TablaSim[a3].valor;

    if(op==PARENTESIS) TablaSim[a1].valor = TablaSim[a2].valor;

    if(op==UMENOS) TablaSim[a1].valor = TablaSim[a2].valor \* -1;

    if(op==POTENCIA) TablaSim[a1].valor = pow(TablaSim[a2].valor, TablaSim[a3].valor);

    if(op==ASIGNAR) TablaSim[a1].valor = TablaSim[a2].valor;

    if(op==SALTAR1){

      if(TablaSim[a1].valor==0) i=a2-1;

    }

    if(op==SALTAR2){

      if(TablaSim[a1].valor==1) i=a2-1;

    }

    if(op==BUCLE\_WHILE) i=a1-2;

    if(op==SALTAR\_FOR){

      if(TablaSim[a1].valor >= TablaSim[a2].valor) i=a3-1;

    }

    if(op==BUCLE\_FOR){

      TablaSim[a1].valor += 1;

      i=a2-1;

    }

  }

}

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);

  if(tok==CADENA) sprintf(TablaSim[nSim].strval,"%s",nom);

  nSim++;

  return nSim - 1;

}

void imprimeTablaSim(){

  int i;

  for(i=0; i<nSim; i++){

    printf("%4d nombre = %6s tok = %6d valor = %4.3lf\n",i,TablaSim[i].nombre,TablaSim[i].token,TablaSim[i].valor);

  }

}

void imprimeTablaCod(){

  int i;

  for(i=0; i<=cx; i++){

    printf("%4d op=%5d  a1=%4d a2=%4d a3=%4d\n",i,TCodigo[i].op,TCodigo[i].a1,TCodigo[i].a2,TCodigo[i].a3);

  }

}

void yyerror(char \*msg){

  printf("Syntax Error: %s on line %i \n", msg, errorLine);

}

int EsPalabraReservada(char lexema[]){

  //strcmp considera mayusculas y minusculas

  //strcasecmp ignora mayusculas de minusculas

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

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

  if(strcasecmp(lexema,"si")==0) return SI;

  if(strcasecmp(lexema,"sino")==0) return SINO;

  if(strcasecmp(lexema,"entonces")==0) return ENTONCES;

  if(strcasecmp(lexema,"mientras")==0) return MIENTRAS;

  if(strcasecmp(lexema,"hacer")==0) return HACER;

  if(strcasecmp(lexema,"calcular")==0) return CALCULAR;

  if(strcasecmp(lexema,"desde")==0) return DESDE;

  if(strcasecmp(lexema,"aumentado")==0) return AUMENTADO;

  if(strcasecmp(lexema,"en")==0) return EN;

  if(strcasecmp(lexema,"hasta")==0) return HASTA;

  if(strcasecmp(lexema,"Imprimir")==0) return SALIDA;

  return ID;

}

int yylex(){

  char c; int i;

  while(1){

    c = getchar();

    if(c == ' ') continue;

    if(c == '\t') continue;

    if(c == '\n'){

      errorLine += 1;

      continue;

    }

    if(c == '"'){

      i = 0;

      do{

        lexema[i++] = c;

        c = getchar();

      }while(c != '"');

      ungetc(c, stdin);

      lexema[i] = '\0';

      return CADENA;

    }

    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);

    }

    return c;

  }

}

int main(){

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

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

  // yyparse();

  printf("Tabla de Simbolos:\n"); imprimeTablaSim();

  printf("Tabla de Codigos\n"); imprimeTablaCod();

  printf("--------------------------------\n");

  InterpretaCodigo();

  printf("--------------------------------\n");

  printf("Tabla de Simbolos despues de interpretar\n"); imprimeTablaSim();

  return 0;

}

Entrada:

Programa Ejemplo

{

Var x=11, y=11, z=2, s=0;

si x < y

[

calcular desde x aumentado en z hasta y

[

s = s + x\*\*2;

]

Imprimir s;

]

sino

[

si x = y

[

Imprimir 0;

]

sino

[

Imprimir "No existe la suma";

]

]

}