```
        { ************************************ }
        { **  Unidad de Ejecucion del Qenqo **  }
        { ************************************ }
        { ** Autor : JOSE LUIS SEGOVIA JUAREZ   }
        { ** Lima - Peru. 1991 ** }

UNIT ejecuni;
interface
uses lexanuni,crt;

procedure ejecuta(tipo:byte);

implementation
PROCEDURE ejecuta;
        { ********************************************* }
        { ** ejecuta las lineas de codigo intermedio ** }
        { ********************************************* }

   VAR c,                    { ** contador de pila ** }
      ncode:INTEGER;
      salir_ejecuta:BOOLEAN;  { * Para salir del modulo de ejecucion * }
      n:INTEGER;              { * numero de tcode en tcode = ncode * }
      ubits : INTEGER;
      numex : INTEGER;
      val1, val2 : REAL;      { * Para dividir el horizonte temporal * }
      datab:STRING[15];
      tinicio, tfinal, tincrem:INTEGER; { * Tiempo inicial, final e increm * }
      xx1, xx2, fx1, fx2, xxx: REAL;    { * Variables auxiliares * }

   PROCEDURE opera(x:INTEGER);
             { ******************************* }
             { * Opera las lineas de codigo  * }
             { ******************************* }
      LABEL salto;
      VAR salida:BOOLEAN;
         y:byte;
         lop1, lop2, lop3 : INTEGER; {* Lugares de los operadores en la tabla de simbolos * }
   BEGIN
      salida:=FALSE;
      WHILE NOT salida DO
      salto:BEGIN lop1:=0; lop2:=0; lop3:=0;
         WITH tcode[x] DO
            IF ((arg1 > 0) AND (arg2 > 0)) THEN
            BEGIN
               IF  (Ts[arg1].objeto  >  0)  AND  (ts[arg1].atr  <>  ID_FUNC)   AND  (ts[arg1].atr  <>
ID_CONTROL)
                  THEN lop1:=trunc(ts[tcode[x].arg1].valor)
               ELSE lop1:=arg1;
               IF  (Ts[arg2].objeto   >  0)  AND  (ts[arg2].atr  <>  ID_FUNC)  AND  (ts[arg2].atr  <>
ID_CONTROL)
                  THEN lop2:=trunc(ts[tcode[x].arg2].valor)
               ELSE lop2:=arg2;
            END;
          IF (tcode[x].arg3 > 0) THEN
            IF (ts[tcode[x].arg3].objeto > 0) AND ((ts[tcode[x].arg3].atr <> ID_FUNC)
               AND (ts[tcode[x].arg3].atr <> ID_CONTROL))
               THEN lop3:=trunc(ts[tcode[x].arg3].valor)
```

```
   ELSE lop3:=tcode[x].arg3;
CASE tcode[x].op OF
RET : BEGIN c:=c-1;
      IF c<=0 THEN BEGIN salida:=TRUE; END
      ELSE BEGIN
         x:=PILA[C+1]; GOTO salto;
      END;
   END;
ASIG: ts[tcode[x].arg1].valor:=ts[lop3(tcode[x].arg3)].valor; {ojo **}
ASGI: BEGIN
      ts[tcode[x].arg1].valor:=TS[tcode[x].arg1].objeto+ts[tcode[x].arg3].valor;
      IF trunc(ts[tcode[x].arg1].valor) > maxts THEN error(1);
      IF ts[trunc(ts[tcode[x].arg1].valor)].atr <> VAL_TABLA THEN ERROR(1);
   END;
ASAT: ts[trunc(ts[tcode[x].arg1].valor)].valor:=ts[lop3].valor;
MAS : ts[tcode[x].arg3].valor :=ts[lop1].valor + ts[lop2].valor;
MENOS: BEGIN
      IF tcode[x].arg1 = 0 THEN
         ts[tcode[x].arg3].valor:=-ts[tcode[x].arg2].valor
      ELSE
         ts[tcode[x].arg3].valor:=ts[lop1].valor-ts[lop2].valor;
   END;
MUL  : ts[tcode[x].arg3].valor:=ts[lop1].valor * ts[lop2].valor;
DIVIS: BEGIN
      IF ts[lop2].valor = 0 THEN error(50) ELSE
         ts[tcode[x].arg3].valor:=ts[lop1].valor / ts[lop2].valor;
   END;
JMP : BEGIN x:=tcode[x].arg3; ncode:=x; GOTO salto; END;

JE  : IF (ts[lop1].valor  = ts[lop2].valor) THEN  x:=tcode[x].arg3-1;
JNE : IF (ts[lop1].valor <> ts[lop2].valor) THEN  x:=tcode[x].arg3-1;
JL  : IF (ts[lop1].valor <  ts[lop2].valor) THEN  x:=tcode[x].arg3-1;
JG  : IF (ts[lop1].valor >  ts[lop2].valor) THEN  x:=tcode[x].arg3-1;
JLE : IF (ts[lop1].valor <= ts[lop2].valor) THEN  x:=tcode[x].arg3-1;
JGE : IF (ts[lop1].valor >= ts[lop2].valor) THEN  x:=tcode[x].arg3-1;
VE  : IF (ts[lop1].valor  = ts[lop2].valor) THEN  ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
VNE : IF (ts[lop1].valor <> ts[lop2].valor) THEN  ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
VL  : IF (ts[lop1].valor <  ts[lop2].valor) THEN  ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
VG  : IF (ts[lop1].valor >  ts[lop2].valor) THEN  ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
VLE : IF (ts[lop1].valor <= ts[lop2].valor) THEN  ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
VGE : IF (ts[lop1].valor >= ts[lop2].valor) THEN  ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
tOR : ts[tcode[x].arg3].valor:=ts[lop1].valor+ts[lop2].valor;
tY  : ts[tcode[x].arg3].valor:=ts[lop1].valor*ts[lop2].valor;
tNO : IF (ts[tcode[x].arg3].valor=0) THEN ts[tcode[x].arg3].valor:=1
   ELSE ts[tcode[x].arg3].valor:=0;
CALL:BEGIN
      CASE tcode[x].arg1 OF
      42 : ts[tcode[x].arg1].valor:=abs(ts[tcode[x].arg1+1].valor);
      44 : ts[tcode[x].arg1].valor:=arctan(ts[tcode[x].arg1+1].valor);
      46 : ts[tcode[x].arg1].valor:=cos(ts[tcode[x].arg1+1].valor);
      48 : ts[tcode[x].arg1].valor:=cos(ts[tcode[x].arg1+1].valor);
      50 : ts[tcode[x].arg1].valor:=cos(ts[tcode[x].arg1+1].valor);
```

```
            52 : ts[tcode[x].arg1].valor:=exp(ts[tcode[x].arg1+1].valor);
            54 : ts[tcode[x].arg1].valor:=frac(ts[tcode[x].arg1+1].valor);
            56 : ts[tcode[x].arg1].valor:=int(ts[tcode[x].arg1+1].valor);
            58 : ts[tcode[x].arg1].valor:=sqr(ts[tcode[x].arg1+1].valor);
            62 : ts[tcode[x].arg1].valor:=ln(ts[tcode[x].arg1+1].valor);
            64 : ts[tcode[x].arg1].valor:=ln(ts[tcode[x].arg1+1].valor)/ln(10);
            66 : ts[tcode[x].arg1].valor:=sqrt(ts[tcode[x].arg1+1].valor);
            68 : ts[tcode[x].arg1].valor:=random;
            70 : ts[tcode[x].arg1].valor:=random;
            72 : ts[tcode[x].arg1].valor:=sin(ts[tcode[x].arg1+1].valor);
            ELSE  BEGIN
                  c:=c+1;
                  pila[c]:=x+1;
                  ubits:=trunc(ts[tcode[x].arg1].objeto);
                  x:=ubits;
                  GOTO salto;
               END;
            END;
         END;
      WRAR     :BEGIN
            numex:=1;
            IF w_arch THEN
               IF tcode[x].arg2 = 0 THEN writeln(archwork)
               ELSE
               BEGIN
                  write(archwork, ts[trunc(tabwrit[tcode[x].arg1+numex-1].valor)].valor:12:2);
                  FOR y:=1 TO nvargraf DO
                  BEGIN
                     IF dat_vargraf[y].nombre=trunc(tabwrit[tcode[x].arg1].valor) THEN
                        IF NOT dat_vargraf[y].flagrango THEN
                        BEGIN
                           IF    (dat_vargraf[y].min     >     ts[trunc(tabwrit[tcode[x].arg1    +
numex-1].valor)].valor) THEN
                              dat_vargraf[y].min      :=      ts[trunc(tabwrit[tcode[x].arg1    +
numex-1].valor)].valor;
                           IF    (dat_Vargraf[y].max     <     ts[trunc(tabwrit[tcode[x].arg1    +
numex-1].valor)].valor) THEN
                              dat_vargraf[y].max:=ts[trunc(tabwrit[tcode[x].arg1                +
numex-1].valor)].valor;
                        END;
                  END;
               END;
         END;
      ELSE error(59);
      END;
      x:=x+1;
   END;
END;

            { ** Principal de ejecuta ** }
BEGIN
   c:=1;
   ininter:=1;
   IF nvargraf > 0 THEN
   BEGIN
      narchwork:=copy(n_archivo,1,pos('.',n_archivo))+'TXT';
      assign(archwork,narchwork);
      rewrite(archwork);
```

```
   END;
   IF tipo=1 THEN BEGIN  opera(ininter); IF nvargraf > 0 THEN CLOSE(ARCHWORK); exit;   END; {salir en
tipo uno}
   ts[39].valor:=ts[10].valor; {inicio de variable tiempo} {controla el tiempo del sistema}
   IF nvargraf > 0 THEN BEGIN
      dat_vargraf[1].min:=ts[iniciog].valor;
      dat_vargraf[1].max:=ts[finalg].valor;
   END;
   WHILE  ts[39].valor <= ts[11].valor DO  { ** tiempo < hasta_t ** }
   BEGIN
      w_arch:=FALSE;
      val1 := round(ts[39].valor*100000);  {maximo de particion de tiempo = /e-5}
      val2 := round(ts[paso].valor*100000);
      IF val2 = 0 THEN BEGIN IF nvargraf > 0 THEN close(archwork); exit; END;
      IF (frac(val1/val2) = 0) AND (nvargraf > 0) THEN w_arch:=TRUE;
      c:=1;
      gotoxy(42,12); write(val1/100000:12:8);
      salir_ejecuta:=FALSE;
      pila[c]:=ininter;        {** inicio de pila **}
      opera(trunc(ts[maxpalres+1].objeto));
      ts[39].valor:=ts[39].valor+ts[9].valor; {** tiempo en 39, dt en 9 **}
   END;
   IF nvargraf > 0 THEN close(archwork);
END;
BEGIN
END.
```

```
    { ********************************************** }
    { ***       Unidad Grafica del Qenqo         *** }
    { ********************************************** }
    { ** Autor : JOSE LUIS SEGOVIA JUAREZ ** }
    { ** Lima-Peru. 1991 ** }

UNIT grafuni;
INTERFACE
USES lexanuni, graph, crt;
TYPE numstring = STRING[10];

            { ** Variables propias de la unidad grafica ** }
VAR graphdriver, graphmode:INTEGER;
   datos_ventana : viewporttype;
   i, j :INTEGER;
   H             : word;      {*  factor de palabra  *}
   puntero_salida: pointer;
   n_maxx, n_maxy: longint;
   maxX, maxY    : INTEGER;   {* longitud de pantalla *}
   X, Y          : INTEGER;   {* ubicacion de pixel   *}
   n_datos       : INTEGER;   {* numero de puntos en x *}
   vx, vy, vlx, vly: INTEGER; {* datos de ventana interior *}
   pasox, pasoy  :REAL;
   caracterleido :CHAR;
procedure ini_grafico;
procedure grafica(archivo:string);
procedure cierragrafico;
procedure grafico_general;
implementation

PROCEDURE Ini_grafico;
           { ***************************** }
           { *** Inicializa graficos  *** }
           { ***************************** }
BEGIN
   Directvideo:=FALSE;
   detectgraph(graphdriver,graphmode);
   graphdriver:=detect;
   initgraph(graphdriver,graphmode,'');
   setgraphmode(getgraphmode);
   IF graphresult <> GROK THEN halt;  {** Error en graficos **}
   maxX:=getmaxX;
   maxY:=getmaxY;
   setusercharsize(1,1,1,1);
   settextstyle(DEFAULTFONT, horizdir, usercharsize);
   SetVisualPage(1);
   SetActivePage(1);
   setviewport(0,0,maxx,maxy,clipon);
END;
PROCEDURE cierragrafico;
BEGIN
   closegraph;
END;

FUNCTION I2S(L:longint):STRING;
      { ******************* }
      { * Entero a String * }
```

```
      { ******************* }
   VAR s:STRING;
BEGIN
   str(L,S); I2S:=S;
END;

FUNCTION R2S(L:REAL;decim:INTEGER):numstring;
      { ********************************** }
      { ** Convierte numero real string ** }
      { ********************************** }
   VAR s:STRING;
BEGIN
   str(L,S);
   r2S:=S;
END;

PROCEDURE grafico_general;
       { ************************************************** }
       { ** Construccion de grafico base y sus coordenadas ** }
       { ************************************************** }
   VAR j:INTEGER; {contador}
      mfactor:REAL;
BEGIN
   H:=TextHeight('M');
   getviewsettings(datos_ventana);
   WITH datos_ventana DO
      setviewport(0,0,maxx,maxy,clipon);
   WITH datos_ventana DO
   BEGIN
      vx:=H+50;
      vy:=H*11+H DIV 2;
      v1x:=(x2-x1)-2*H-H DIV 2;
      v1y:=(y2-y1)-2*H;
      pasoy:=((y2-y1)-h*11) DIV 20;
      pasox:=((x2-x1)-2*H-50) DIV 40;
      line(H+50,        11*H,      H+50, (y2-y1)-H*2);
      line(H+50, (y2-y1)-H*2, (x2-x1)-2*H, (y2-y1)-H*2);
      J:=(y2-y1)-H*2;
      outtextxy(1, 1, ' * QENQO *    RESULTADOS      ');
      outtextxy(1, 2*h+2, ' MODELO :');
      outtextxy(10*h, 2*h+2, ts[maxpalres+1].nombre);
      outtextxy(x2-42*H, 1, 'LINEA  VARIABLE   FACTOR  V.MINIM V.MAXIM');
      line(1, h+4, x2, h+4);
      line(x2-42*h-4, 1, x2-42*h-4, H*10);
      line(x2-35*h-4, 1, x2-35*h-4, h*10);
      line(x2-26*h-4, 1, x2-26*h-4, h*10);
      line(x2-16*h-4, 1, x2-16*h-4, h*10);
      line(x2-8*h-4,  1, x2-8*h-4,  h*10);
      line(1, h*10, x2, h*10);
      FOR i:=0 TO n_maxy DO
      BEGIN
         line(H+45, j, H+50, j);
         IF (i/2 - i DIV 2) = 0 THEN outtextxy(4*H, j-3,  I2s(i DIV 2));
         j:=trunc(j-pasoy);
      END;
      j:=h+50;
      FOR i:=0 TO n_maxx DO
      BEGIN
```

```
        line(j, (y2-y1)-2*H, j, (y2-y1)-H-h DIV 2);
        IF frac(i/4) = 0 THEN
           line(j, (y2-y1)-2*H, j, (y2-y1)-H);
        j:=trunc(j+pasox);
      END;
        {* escribir las variables en pantalla *}
      FOR j:=2 TO nvargraf DO   {ojo sin +1}
      BEGIN
        setlinestyle(dat_Vargraf[j].tipo_linea, 0, 1);
        line(x2-41*H-3, j*H+4, x2-36*H, j*H+4);
        outtextxy(x2-35*H, (j)*H+1, ts[dat_vargraf[j].nombre].nombre);
        IF abs(dat_vargraf[j].max-dat_vargraf[j].min) <= 0.0001 THEN
           dat_vargraf[j].factor:=1
        ELSE dat_vargraf[j].factor := (v1y-vy)/(dat_vargraf[j].max - dat_vargraf[j].min) ;
                  { #pixels/dato}
        mfactor:=(dat_vargraf[j].max - dat_vargraf[j].min) / 10;
        IF (mfactor < 1.5) AND (mfactor > -1.5)
           THEN mfactor:=1;
        outtextxy(x2-26*H, j*H+1, i2s(round(mfactor)));
        outtextxy(x2-16*H, j*H+1, i2s(round(dat_vargraf[j].min)));
        outtextxy(x2-08*H, j*H+1, i2s(round(dat_vargraf[j].max)));
      END;
    END;
END;

PROCEDURE grafica(archivo:STRING);
                                {************************}
                                {*** grafico general ***}
                                {************************}
   PROCEDURE plotear; {lee una linea de archivo, cambia coordenas y pone en pantalla}
      VAR i, j:INTEGER;
         dtx :REAL;
         valores:ARRAY[1..maxvar] OF REAL;
         xx, yy :INTEGER;
   BEGIN
      i:=0;
      WITH datos_ventana DO
         WHILE NOT eof(archwork) DO
         BEGIN
            read(archwork,dtx); {dtx inicio de grafico}
            FOR j:=2 TO nvargraf DO read(archwork,valores[j]);
            readln(archwork);
            xx                          :=trunc((dtx-dat_vargraf[1].min)                    *
(v1x-vx)/trunc(dat_vargraf[1].max-dat_vargraf[1].min));
            FOR j:=2 TO nvargraf DO
            BEGIN
               yy:=trunc((v1y-vy)-(valores[j]-dat_vargraf[j].min)*dat_vargraf[j].factor);
               setlinestyle(dat_Vargraf[j].tipo_linea, 0, 1);
               IF i >= 2 THEN line(dat_vargraf[j].x,dat_vargraf[j].y, xx, yy);
               dat_vargraf[j].x:=xx;
               dat_vargraf[j].y:=yy;
            END;
            i:=i+1;
         END;
      setlinestyle(0,0,1);
   END;

   PROCEDURE grafico_resultados;
```

```
                        {*****************************}
                        {* Ordena los graficos       *}
                        {*****************************}
  BEGIN
    reset(archwork);
    WITH datos_ventana DO
    BEGIN
       outtextxy(x2-11*H, (y2-y1)-4*H, ts[dat_vargraf[1].nombre].nombre);
       j:=h+50;
       dat_vargraf[1].factor:=(dat_vargraf[1].max - dat_vargraf[1].min) / n_maxx;
       settextjustify(centertext, toptext);
       FOR i:=0 TO n_maxx DO
       BEGIN
          IF frac(i/4) = 0 THEN
             outtextxy(x1+j, (y2-y1)-H, i2s(trunc(dat_vargraf[1].min+i*dat_vargraf[1].factor)));
          j:=trunc(j+pasox);
       END;
       settextjustify(lefttext, centertext);
       setviewport(vx, vy-(h DIV 3), v1x, v1y+(h DIV 3), clipon);
       plotear;
    END;
    close(archwork);
  END;
BEGIN              {*** Programa principal Grafica *** }
  n_datos:=0;
  n_maxx:=40;
  n_maxy:=20;
  setviewport(1,7*h+2,35*H,9*h,clipon);
  clearviewport;
  WITH datos_ventana DO
     setviewport(0,0,maxx,maxy,clipon);
  outtextxy(1, 8*h+2, ' Espere ...                              ');
  grafico_resultados;
  setviewport(1,7*h+2,35*H,9*h,clipon);
  clearviewport;
  WITH datos_ventana DO
     setviewport(0,0,maxx,maxy,clipon);
  outtextxy(1, 8*h+2, ' Presione <ENTER> para salir al menu.');
END;
BEGIN
END.
```

```
    { ************************************** }
    { **   Analizador Sintactico de Qenqo  *** }
    { ************************************** }
    { ** Lima - Peru *. 1991 **}

UNIT parseuni;
INTERFACE
USES lexanuni;
procedure gencode(vop, varg1, varg2, varg3:integer);
Procedure genvararray(tabla:integer; init, fint, inct:real; inicio, fin:integer);
Procedure genvartemp;
procedure parser;
implementation
PROCEDURE gencode(vop, varg1, varg2, varg3:INTEGER);
    {* Crea lineas del Codigo Intermedio ** }
BEGIN
   cx:=cx+1;
   tcode[cx].op   := vop;
   tcode[cx].arg1 := varg1;
   tcode[cx].arg2 := varg2;
   tcode[cx].arg3 := varg3;
   IF cx > maxcode THEN error(78);
END;

PROCEDURE genvararray(tabla:INTEGER; init, fint, inct:REAL; inicio, fin:INTEGER);
      {** Genera variables de trabajo para ser usados en Tabla de Simbolos **}

   VAR temp1,temp3:STRING[15]; yy:INTEGER;
BEGIN
   temp1:=''; temp3:='';
   str(tabla, temp1);
   temp1:= concat('_T',temp1,'_I');
   varx:=instala_ident(temp1);
   ts[varx].valor:=init;
   str(tabla, temp1);
   temp1:= concat('_T',temp1,'_F');
   varx:=instala_ident(temp1);
   ts[varx].valor:=fint;
   str(tabla, temp1);
   temp1:= concat('_T',temp1,'_C');
   varx:=instala_ident(temp1);
   ts[varx].valor:=inct;
   FOR yy:=inicio TO fin DO
   BEGIN
      str(tabla, temp1);
      str(yy,temp3);
      temp1:= concat('_T',temp1,'_',temp3);
      varx := instala_ident(temp1);
   END;
END;

PROCEDURE genvartemp;  { ** Genera variables temporal ** }
   VAR temp1:STRING[15];
BEGIN
   vtn := vtn+1;  temp2:=' '; temp1:='';
   str(vtn,temp1);
   temp1:= concat('_T',temp1);
```

```
   varx := instala_ident(templ);
END;

{ ******************************************************** }
{ *************    Analizador Sintactico ***************** }
{ ******************************************************** }

PROCEDURE parser;
   PROCEDURE expr; FORWARD;
   PROCEDURE constante;      { ** Analiza las Constantes ** }

      VAR t23:INTEGER; varx23:INTEGER;
   BEGIN
      varx23:=varx;
      tok:=sigtok;
      IF tok = APOSTROFO THEN
      BEGIN
         tok:=sigtok;
         IF tok <> LITERAL THEN error(5);
         tok:=sigtok;
         IF tok <> APOSTROFO THEN error(5);
         exit;
      END;
      IF (tok = MAS) OR (tok = MENOS) THEN tok:=sigtok;
      IF (ts[lugar].atr = ID_CONS) OR (ts[lugar].atr = ID_CONTROL) THEN
      BEGIN
         NUMERO:=TRUE;
         varx:=lugar;
         exit;
      END;
      IF (tok = NUM_I) OR (tok = NUM_R) THEN
      BEGIN
         numero:=TRUE;
         varx:=lugar; exit;
      END
      ELSE error(32);
   END;

   PROCEDURE parmlista(lugar1:INTEGER);
    { ************************************ }
    { ** Analiza la Lista de Parametros ** }
    { ************************************ }
      VAR salida:BOOLEAN; x,xtot:INTEGER;
         tabtipo:ARRAY[1..20] OF byte;
   BEGIN
      fillchar(tabtipo,sizeof(tabtipo),0);
      salida:=FALSE;
      x:=0; xtot:=0;
      tok:=sigtok;
      IF tok = PUNTO_COMA  THEN
      BEGIN
         reponetok;
         exit;
      END;
      IF tok <> Par_I THEN
      BEGIN
         Reponetok;
         error(63);
```

```
      exit;
    END;
    WHILE NOT salida DO
    BEGIN
      REPEAT
        tok := sigtok;
        IF (tok <> IDENT) THEN error(2);
        x:=x+1; xtot := xtot + 1;
        tabtipo[x]:=lugar;
        tok:=sigtok;
      UNTIL tok <> coma;
      IF (tok <> DOS_PUNTOS_V) THEN error(61);
      tok:=sigtok;
      IF (ts[LUGAR].atr <> ID_TIPO) OR (ts[lugar].objeto = tTABLA) THEN error(12);
      WHILE x >= 1 DO
      BEGIN
      ts[tabtipo[x]].atr:=tok; x:=x-1;  END;          {** Tipo **}
      tok:=sigtok;
      IF tok=PAR_D THEN SALIDA:=TRUE;
    END;
    ts[lugar1].valor := xtot;
    IF Tok = PAR_D THEN exit
    ELSE error(64);
  END;

  FUNCTION variable:INTEGER;
   { ************************************ }
   { ** Analiza las variables          ** }
   { ************************************ }

    VAR txx, vartabla:INTEGER;
  BEGIN
    IF (ts[lugar].atr = ID_TABLA) AND (tok <> NUM_R) THEN
    BEGIN
      vartabla:=trunc(ts[lugar].valor);
      tok:=sigtok;
      IF (tok <> CORCH_I) THEN error(67);
      genvartemp;
      txx:=varx;
      ts[txx].objeto:=vartabla+2;
      expr;
      gencode(ASGI, txx, 0, varx);
      tok:=sigtok;
      IF (tok <> CORCH_D) THEN error(68);
      variable:=txx; {lugar donde esta el desplazamiento, en valor}
    END
    ELSE variable:=lugar;  {modificar sus valores anteriores y posteriores}
  END;

  PROCEDURE factor;
   { ************************************ }
   { ** Analiza la sintaxis de Factor  ** }
   { ************************************ }
    LABEL otraexpr;
    VAR i,j, nparm:INTEGER;
  BEGIN
    tok:=sigtok;
    IF es_variable(lugar) AND (tok = IDENT) THEN
```

```
 BEGIN
    IF (es_variable(lugar) AND flagnivel) AND
       NOT (ts[lugar].atr=id_flujo) THEN error(8);
    ubivarfact:=lugar;
    varx:=variable;  {en ts}
    exit;
 END;
 IF tok = PAR_I THEN
 BEGIN
    expr;
    tok:=sigtok;
    IF tok<>PAR_D THEN error(64);
    exit;
 END;
 { ** Analiza Funciones ** }

if (ts[lugar].atr = ID_FUNC) and (tok <> NUM_I) then
   begin
     i:=lugar;
     nparm:=0;
     tok:=sigtok;
     if (tok <> PAR_I) and (ts[i].valor>0) then error(63);
     if (tok = PAR_I) and (ts[i].valor>0) then
     begin
       repeat
         otraexpr:       expr;
           nparm:=nparm+1;
           j:=i+nparm;
           if (ts[j].atr <> tAUX) {and (ts[i].atr <> ID_func)} then
                 if (ts[j].atr <> ts[varx].atr) then error(26);
           gencode(ASIG, j, 0, varx);
           tok:=sigtok;
           if tok = COMA then
             begin
                goto otraexpr;
             end;
           if (tok <> COMA) then
              if tok <> PAR_D then error(5);
       until tok = PAR_D;
      end
     else reponetok;
     gencode(CALL,i,0,0);
     genvartemp;
     gencode(asig, varx, 0 ,i);
     if nparm <> ts[i].valor then error(22);
     exit;
   end;
  reponetok;
  constante;
END;

PROCEDURE term;
 { ************************************ }
 { ** Analiza la sintaxix de terminos ** }
 { ************************************ }
  VAR salir, flagmd:BOOLEAN; i1,i2, tt: INTEGER;
BEGIN
  i1:=ubivarfact;
```