```
            NUM2 = NUM;
            K = K + 1;
            GOTO BB;
         END;
         CADENA_LEIDA = CADENA_LEIDA!!ASCII(LENGTH(CHAR(NUM1))+1)!!
         CHAR(NUM1)!!ASCII(LENGTH(CHAR(NUM2))+1)!!CHAR(NUM2)!!
         ASCII(LENGTH(CHAR(NUM))+1)!!CHAR(NUM);
      END;
      IF (DIRE_PANTA = 2 | DIRE_PANTA = 15) THEN DO;
         UBICACION(3) = 9;
         DO I = 1 TO 3;
            CALL NORMALIZA;
         END;
         CADENA_LEIDA = CADENA_LEIDA!!ASCII(LENGTH(CLAVE1)+1)!!CLAVE1;
      END;
      IF (DIRE_PANTA = 3 | DIRE_PANTA = 14)THEN DO;
         I = 1;
         CALL NORMALIZA;
         CADENA_LEIDA = CADENA_LEIDA!!ASCII(LENGTH(CLAVE1)+1)!!CLAVE1;
      END;
      CADENA_LEIDA= CADENA_LEIDA!!ASCII(255);
   END;

/*************************************************************************
                              SUBRUTINAS
 *************************************************************************/

ASIGNA: PROCEDURE;

         IF BA_COM_GR = 1 THEN
            CADENA_LEIDA = CADENA_LEIDA !! ASCII(253) !! ASCII(POS);
         CADENA_LEIDA = CADENA_LEIDA !! ASCII(LENGTH(VALOR)+1) !! VALOR;

END;     /*        ASIGNA      */

REASIGNA: PROCEDURE;

            LON = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
            IF LENGTH(VALOR) = 0 THEN DO;
               POS1 = POS1 + LON;
               GOTO XX;
            END;
            IF POS1 = 1 THEN
               CADENA_LEIDA = ASCII(LENGTH(VALOR)+1)!!VALOR!!SUBSTR(CADENA_LEIDA,L
ON+POS1);
            ELSE
               CADENA_LEIDA = SUBSTR(CADENA_LEIDA,1,POS1-1)!!ASCII(LENGTH(VALOR)+1
)!!VALOR!!SUBSTR(CADENA_LEIDA,POS1+LON);
            POS1 = POS1 + LENGTH(VALOR) + 1;
            XX:

END;        /*       REASIGNA       */
/*************************************************************************/

BORRA_RESTO_LINEA: PROCEDURE;
      IF (MODIFICAR="S" & VALIDO = "S" & LENGTH(VALOR) ^= 0) THEN
    CALL BORRA_INFORMACION(LENGTH(VALOR),LONGITUD,X,Y,JJ);
    ELSE DO;
       IF JJ = 1 THEN DO;
          JJ = 0;
    CALL BORRA_INFORMACION(LENGTH(VALOR),LONGITUD,X,Y,JJ);
       END;
   END;
       /* X = X + LENGTH(VALOR);
       DO K = X TO  LONGITUD;
```

```
            CALL GOTOXY(K,Y);
            PUT FILE(OUT) EDIT(" ")(A(1));
         END;
      END;
      A = LONGITUD;
      B = LENGTH(VALOR);
      IF LENGTH(VALOR_V) > B THEN DO;
         DO K = 1 TO (A - B + 1);
            CALL GOTOXY(X + B + K - 1,Y);
            PUT FILE(OUT) EDIT(" ")(A(1));
         END;
      END;
   END;
      DO II = (LONGITUD-LONGITUD_1) TO LONGITUD+1;
         CALL GOTOXY(X+II-1,Y);
         PUT FILE(OUT) EDIT(" ")(A(1));
      END;
      JJ = 0;
   END;      */
   END;        /*      BORRA_RESTO_LINEA          */

   NORMALIZA: PROCEDURE;

   /*     PARA GUARDAR LA CLAVE NORMALIZADA     */

   IC = 0;
   POS = 1;
   DO WHILE(IC < UBICACION(I));
      PG = POS;
      LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
      VALOR = SUBSTR(CADENA_LEIDA,POS+1,LON-1);
      POS = POS + LON;
      IC = IC + 1;
      IF ((DIRE_PANTA = 15 | DIRE_PANTA = 2) & IC = 9 & VALOR = "") THEN DO;
         CADENA_LEIDA = SUBSTR(CADENA_LEIDA,1,PG-1) ||
         ASCII(LENGTH(CLAVE1)+1) || CLAVE1 ||
         SUBSTR(CADENA_LEIDA,POS);
         GOTO FIN;
      END;
   END;
   LON1 = RANK(SUBSTR(CADENA_LEIDA,POS,1));
   LON1 = LON1 - 1;
   IF LON1 = 0 THEN DO;
      IF VALOR = "" THEN
         GOTO FIN;
      CLAVE = VALOR;
      CALL CORRIGE(CLAVE,K);
      IF UBICACION(I) = 1 THEN
         CADENA_LEIDA=ASCII(LENGTH(CLAVE)+1)||CLAVE||SUBSTR(CADENA_LEIDA,POS);
      ELSE
         CADENA_LEIDA=SUBSTR(CADENA_LEIDA,1,PG-1)||ASCII(LENGTH(CLAVE)+1)||
         CLAVE||SUBSTR(CADENA_LEIDA,POS);
   END;
   ELSE DO;
      CLAVE = VALOR||" "||SUBSTR(CADENA_LEIDA,POS+1,LON1);
      CALL CORRIGE(CLAVE,K);
      LON2 = LENGTH(CLAVE);
      POS = POS + LON1 + 1;
      IF LON2 <= 70 THEN DO;
         IF UBICACION(I) = 1 THEN
            CADENA_LEIDA = ASCII(LON2+1)||CLAVE||ASCII(1)||SUBSTR(CADENA_LEIDA,POS)
   ;
         ELSE
            CADENA_LEIDA = SUBSTR(CADENA_LEIDA,1,PG-1)||ASCII(LON2+1)||
```

```
            CLAVE!!ASCII(I)!!SUBSTR(CADENA_LEIDA,POS);
      END;
      ELSE DO;
         VALOR = SUBSTR(CLAVE,K);
         CLAVE = SUBSTR(CLAVE,1,K-1);
         IF UBICACION(I) = 1 THEN
            CADENA_LEIDA = ASCII(LENGTH(CLAVE)+1)!!CLAVE!!
            ASCII(LENGTH(VALOR)+1)!!VALOR!!SUBSTR(CADENA_LEIDA,POS);
         ELSE
            CADENA_LEIDA = SUBSTR(CADENA_LEIDA,1,PG-1)!!ASCII(LENGTH(CLAVE)+1)!!
            CLAVE!!ASCII(LENGTH(VALOR)+1)!!VALOR!!SUBSTR(CADENA_LEIDA,POS);
      END;
   END;

   FIN:

   END;    /*    NORMALIZA     */

   FIN10:
   END; /* CREA_INFORMACION      */
```

```
POS_RE = 0;
/*IF IG ^= 1 THEN DO;
   DO WHILE(POS < IG);
      UP:
      IF VECTOR_COOR(POS) = ASCII(253) THEN DO;
         POS = POS + 5;
         GOTO FIN_W;
      END;
      IF VECTOR_COOR(POS) = ASCII(254) THEN DO;
         POS = POS + 1;
         GOTO UP;
      END;
      POS1 = POS1 + RANK(SUBSTR(CADENA_LEIDA,POS1,1));
      IF SUBSTR(CADENA_LEIDA,POS1,1) = ASCII(253) THEN
         POS1 = POS1 + 2;
      IF SUBSTR(CADENA_LEIDA,POS1,1) = ASCII(254) THEN
         POS1 = POS1 + 1;
      POS = POS + 5;
      FIN_W:
   END;
END; */
DO WHILE(SUBSTR(CADENA_LEIDA,POS1,1) ^= ASCII(255) & CL ^= FG);
   IF VECTOR_COOR(POS) = ASCII(253) THEN DO;
      CR = 0;
      POS_RE = RANK(VECTOR_COOR(POS+1));
      POS = POS + 5;
   END;
   X = RANK(VECTOR_COOR(POS));
   Y = RANK(VECTOR_COOR(POS+1));
   IF POS_RE ^= 0 THEN DO;
      Y = Y + POS_RE*CR;
      CR = CR + 1;
   END;
   LON = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
   VALOR = SUBSTR(CADENA_LEIDA,POS1+1,LON-1);
   CALL GOTOXY(X,Y);
   PUT FILE(OUT) EDIT(VALOR)(A);
   POS1 = POS1 + LON;
   POS = POS + 5;
   CL = CL + 1;
   VALOR = "";
END;

END; /*       RECONSTRUYE          */
```

```
PRIMERO: PROCEDURE;

/*****************************************************************************
 *                                                                           *
 *    PROGRAMA ENCARGADO DE CREAR,MODIFICAR,ELIMINAR Y                       *
 *    CONSULTAR BANCOS Y MONEDAS                                             *
 *                                                                           *
 *    PARAMETROS:                                                            *
 *    LLAMADA POR:                                                           *
 *    SELECCION                                                              *
 *                                                                           *
 *    LLAMA A:                                                               *
 *    GENERA                                                                 *
 *    GOTOXY                                                                 *
 *                                                                           *
 *    VARIABLES:                                                             *
 *    NOMBRE: VARIABLE QUE IDENTIFICA AL BANCO Y LA MONEDA                   *
 *    TCOM: TABLA DE COMANDOS                                                *
 *    CONTROL2: ESTRUCTURA DE DATOS QUE CONTIENE LA DIRECCION                *
 *    DEL ULTIMO REGISTRO OCUPADO EN EL ARCHIVO DE BANCOS                    *
 *    CONTROL1: ESTRUCTURA DE DATOS QUE CONTIENE LA DIRECCION                *
 *    DEL ULTIMO REGISTRO OCUPADO EN EL ARCHIVO DE MONEDAS                   *
 *    INFMON: ESTRUCTURA DE DATOS QUE CONTIENE LA SIGUIENTE                  *
 *    INFORMACION:                                                           *
 *    BL_A: BLOQUE ANTERIOR                                                  *
 *    BL_S: BLOQUE SUGUIENTE                                                 *
 *    CONT_M: CANTIDAD DE MONEDAS EN EL BLOQUE                               *
 *    SITUACION_M: SITUACION DE LA MONEDA                                    *
 *    MONE: VARIABLE QUE CONTIENE LOS NOMBRES DE LA MONEDA                   *
 *    SIMBOLO: VARIABLE QUE CONTIENE EL SIMBOLO DE LA MONEDA                 *
 *    INFBAN: ESTRUCTURA DE DATOS QUE CONTIENE LA SIGUIENTE                  *
 *    INFORMACION:                                                           *
 *    BLA: BLOQUE ANTERIOR                                                   *
 *    BLS: BLOQUE SUGUIENTE                                                  *
 *    CONT_B: CANTIDAD DE BANCOS  EN EL BLOQUE                               *
 *    SITUACION_M: SITUACION DEL BANCO:                                      *
 *    E:  ELIMINADO                                                          *
 *    NOMBRE_B : VARIABLE QUE CONTIENE EL NOMBRE DEL BANCO                   *
 *    DIR_B: VARIABLE QUE CONTIENE LA DIRECCION DEL BANCO                    *
 *                                                                           *
 *****************************************************************************/

DCL
(NOMBRE10,CK)                                          CHAR(1) VAR,
(LINEA(2),NOMBRE3)                                     CHAR(1),
(PASADA,BANDERA,REPITA,MOD,EL,ENCONTRADO)                 BIT(1),
(LON,L,K,COM,LLEVO,DIR,TOT,T,CONT,BLOV2,CONT_BL,DIR1,T,
 BLOV1,DIRECCION,CONT_M1,J,DIRECCION1)                 BIN FIXED,
(BANCOS,MONEDAS,IN,OUT,IMP,PANTALLAS,COORDENADAS)      FILE,
(NOMBRE,DATO1)                                         CHAR(8) VAR,
(NOMBRE1,DATO)                                         CHAR(30) VAR,
NOMBRE2                                                CHAR(120) VAR,
01 CONTROL1,
   02 MAX_DIR1                                         BIN FIXED,
01 INFMON,
   02 BL_A                                             BIN FIXED,
   02 BL_S                                             BIN FIXED,
   02 CONT_M                                           BIN FIXED,
   02 SITUACION_M(12)                                  CHAR(1) VAR,
   02 MONE(12)                                         CHAR(10) VAR,
   02 SIMBOLO(12)                                      CHAR(5) VAR;
DCL
01 CONTROL2,
```

```
                                                        BIN FIXED,

   02 MAX_DIR2                                          BIN FIXED,
01 INFRAN,                                              BIN FIXED,
   02 BLA                                               BIN FIXED,
   02 BLS                                               BIN FIXED,
   02 CONT_B                                            CHAR(1),
   02 NUMERO_B(3)                                       CHAR(30) VAR,
   02 SITUACION(3)                                      CHAR(120) VAR,
   02 NOMBRE_B(3)
   02 DIR_B(3)
GOTOXY ENTRY(BIN FIXED,BIN FIXED),
GENERA ENTRY(BIN FIXED,BIN FIXED,BIN FIXED);
DCL
    TCOM(7,2) CHAR(1) STATIC INITIAL(
    "C","R",
    "F","L",
    "M","O",

    RECONSTRUYE: PROCEDURE(DIRE_PANTA,DIRE_COR,IG,FG,IV,FV,CADENA_LEIDA);

    /*************************************************************************
    *                                                                        *
    *     PROGRAMA QUE SE UTILIZA PARA LA RECUPERACION DE PANTALLAS          *
    *                                                                        *
    *     PARAMETROS:                                                        *
    *                                                                        *
    *          DIRE_PANTA: CONTIENE LA DIRECCION DE LA PANTALLA EN           *
    *          EL ARCHIVO PANTALLAS                                          *
    *                                                                        *
    *          DIRE_COR: CONTIENE LA DIRECCION DE LAS COORDENADAS DE         *
    *          LOS DATOS EN EL ARCHIVO COORDENADAS                           *
    *                                                                        *
    *          IG: INICIO DE COORDENADAS EN EL VECTOR                        *
    *                                                                        *
    *          FG: FIN DE COORDENADAS                                        *
    *                                                                        *
    *          IV: INICIO DE LA VENTANA DONDE COMIENZA LOS CAMPOS            *
    *          REPETITIVOS                                                   *
    *                                                                        *
    *          FV: FIN DE LOS CAMPOS REPETITIVO                              *
    *                                                                        *
    *          CADENA_LEIDA: CONTIENE LA INFORMACION A RECONSTRUIR           *
    *                                                                        *
    *     VARIABLES:                                                         *
    *                                                                        *
    *     LLAMADA POR:                                                       *
    *                                                                        *
    *          BUSCA_AU_PROV_PP                                              *
    *                                                                        *
    *          BUSCA_ACTIVIDAD                                               *
    *                                                                        *
    *          CONSULTA_OBSERVACION                                          *
    *                                                                        *
    *          MAS_MENOS_LINEAS                                              *
    *                                                                        *
    *     LLAMA A:                                                           *
    *                                                                        *
    *          GENERA                                                        *
    *                                                                        *
    *          GOTOXY                                                        *
    *                                                                        *
    *************************************************************************/
```

```
      PUT FILE(OUT) PAGE;
      CALL GENERA(21,1,100);
      I = 1;
      IF CONT_R ^= 0 THEN DO;
         DO K = 1 TO CONT_R;
            CALL GOTOXY(14,I);
            PUT FILE(OUT) EDIT(SITUACION(K))(A);
            I = I + 1;
            CALL GOTOXY(14,I);
            PUT FILE(OUT) EDIT(NUMERO_R(K))(A);
            I = I + 1;
            CALL GOTOXY(14,I);
            PUT FILE(OUT) EDIT(NOMBRE_R(K))(A);
            I = I + 1;
            CALL GOTOXY(14,I);
            PUT FILE(OUT) EDIT(DIR_R(K))(A);
            IF K = 3 THEN
               I = I + 5;
            ELSE
               I = I + 3;
         END;
       END;
     END;
     ELSE DO;
       READ FILE(MONEDAS) INTO(INFMON) KEY(1);
       DIRECCION = 1;
ADE1: PUT FILE(OUT) PAGE;
       CALL GOTOXY(6,0);
PUT FILE(OUT) EDIT(ASCII(29),"MONEDA","SIMBOLO",'SITUACION',ASCII(28))(A,A(6),X(
8),A(7),X(4),A(9),A);
       IF CONT_M ^= 0 THEN DO;
          DO K = 1 TO CONT_M;
PUT FILE(OUT) SKIP EDIT(MONE(K),SIMBOLO(K),
SITUACION_M(K))(X(6),A(10),X(8),A(6),X(4),A(1));
          END;
       END;
     END;
REPETIR:
      CALL GOTOXY(0,21);
      PUT FILE(OUT) EDIT("?:")(A);
      PUT FILE(OUT) SKIP EDIT("CREAR,ELIMINAR,MODIFICAR,+,-,IMPRIMIR,FIN","(",")
)(A,X(27),A,X(2),A);
      CALL GOTOXY(69,22);
      GET FILE(IN) SKIP EDIT((LINEA(I) DO I = 1 TO 2))(A(1));
      IF (LINEA(1) = "" & LINEA(2) = "") THEN GOTO REPETIR;
      CALL GOTOXY(0,23);
      PUT FILE(OUT) EDIT(CK)(A);
      K = 1;
      COM = 0;
      DO WHILE(COM = 0 & K <= 7);
         IF (TCOM(K,1) = LINEA(1) & TCOM(K,2) = LINEA(2)) THEN
            COM = K;
         ELSE
           K = K + 1;
       END;
      IF COM=0 THEN CALL ERROR;
      ELSE
        DO CASE(COM);
           CALL CREAR;
           CALL ELIMINAR;
           CALL MODIFICAR;
           CALL IMPRIMIR;
           CALL ADELANTE;
           CALL RETROCEDE;
```

```
  END;
IF COM=7 THEN DO;
   PUT FILE(OUT) PAGE;
 GOTO PRINCIPIO;
END;
ELSE GOTO REPETIR;

ERROR: PROCEDURE;
       CALL GOTOXY(0,23);
       PUT FILE(OUT) EDIT("ERROR EN EL COMANDO")(A);
       GOTO REPETIR;
END;/* ERROR */


/*  RUTINA QUE CREA UN NUEVO BANCO Y/O MONEDA */
CREAR: PROCEDURE;
  REPITA = "1"B;
  IF BANDERA = "1"B THEN DO;
    IF BLS ^= 0 THEN DO;
       DO WHILE(BLS ^= 0);
          DIR = BLS;
          READ FILE(BANCOS) INTO(INFBAN) KEY(DIR);
       END;
       PUT FILE(OUT) PAGE;
       CALL GENERA(21,1,100);
       I = 1;
       DO K = 1 TO CONT_B;
          CALL GOTOXY(14,I);
          PUT FILE(OUT) EDIT(SITUACION(K))(A);
          I = I + 1;
          CALL GOTOXY(14,I);
          PUT FILE(OUT) EDIT(NUMERO_B(K))(A);
          I = I + 1;
          CALL GOTOXY(14,I);
          PUT FILE(OUT) EDIT(NOMBRE_B(K))(A);
          I = I + 1;
          CALL GOTOXY(14,I);
          PUT FILE(OUT) EDIT(DIR_B(K))(A);
          IF K = 3 THEN
             I = I + 5;
          ELSE
             I = I + 3;
       END;
    END;
     IF CONT_B = 3 THEN DO;
           I = 0;
          CONT = 1;
          PUT FILE(OUT) PAGE;
          CALL GENERA(21,1,100);
          I = 3;
          DO WHILE(REPITA = "1"B & CONT < 4);
             CALL GOTOXY(14,I);
             I = I + 1;
             READ FILE(IN) INTO(NOMBRE1);
             IF NOMBRE1 = "" THEN DO;
                REPITA = "0"B;
                J = I-1;
                GOTO AL;
             END;
             CALL VERIFICA_BANCO;
             IF I = 1 THEN DO;
                CALL PIDE_VACIO_B2;
                BLS = BLOQ2;
```

```
            REWRITE FILE(BANCOS) FROM(INFBAN) KEY(DIR);
            BLA = DIR;
            BLS = 0;
            DIR = BLOV2;
         END;
         CONT_B = 1;
         NUMERO_B(1) = NUMERO_B(3) + 1;
         NOMBRE_B(1) = NOMBRE1;
         T = T + 1;
         CALL GOTOXY(14,T);
         READ FILE(IN) INTO(NOMBRE2);
         DIR_B(1) = NOMBRE2;
         IF T = 1 THEN
            REWRITE FILE(BANCOS) FROM(INFBAN) KEY(DIR);
         T = T + 5;
         AL:
         CONT = CONT + 1;
         TA:
      END;
      IF T ^= 0 THEN DO;
         CONT_B = 1;
         WRITE FILE(BANCOS) FROM(INFBAN) KEYFROM(DIR);
      END;
    GOTO REPETIR;
  END;
  IF CONT_B = 2 THEN DO;
 TE: CALL GOTOXY(14,15);
     READ FILE(IN) INTO(NOMBRE1);
     IF NOMBRE1 = "" THEN GOTO FINAL;
     CALL VERTICA_BANCO;
     NUMERO_B(3) = NUMERO_B(2) + 1;
     NOMBRE_B(3) = NOMBRE1;
     CALL GOTOXY(14,16);
     READ FILE(IN) INTO(NOMBRE2);
     DIR_B(3) = NOMBRE2;
     CONT_B = 3;
     REWRITE FILE(BANCOS) FROM(INFBAN) KEY(DIR);
     GOTO FINAL;
  END;
  IF CONT_B = 1 THEN DO;
     CONT = 2;
     T = 9;
     DO WHILE(REPITA = "1"B & CONT < 4);
     CALL GOTOXY(14,T);
     READ FILE(IN) INTO(NOMBRE1);
     IF NOMBRE1 = "" THEN DO;
        REPITA = "0"B;
        GOTO EL;
     END;
     CALL VERTICA_BANCO;
     CONT_B = CONT_B + 1;
     NUMERO_B(CONT_B) = NUMERO_B(CONT_B-1) + 1;
     NOMBRE_B(CONT_B) = NOMBRE1;
     T = T + 1;
     CALL GOTOXY(14,T);
     READ FILE(IN) INTO(NOMBRE2);
     DIR_B(CONT_B) = NOMBRE2;
     T = T + 5;
     EL:
     CONT = CONT + 1;
     IT:
   END;
   IF CONT_B ^= 1 THEN
```

```
                REWRITE FILE(BANCOS) FROM(INFBAN) KEY(DIR);
                GOTO FINAL;
              END;
              IF CONT_B = 0 THEN DO;
                 CONT = 1;
                 T = 0;
                 PUT FILE(OUT) PAGE;
                 CALL GENERA(21,1,100);
                 T = 1;
                 DO WHILE(REPITA = '1'B & CONT < 4);
                    CALL GOTOXY(14,T);
                    T = T+1;
                    READ FILE(IN) INTO(NOMBRE1);
                    IF NOMBRE1 = '' THEN DO;
                       REPITA = '0'B;
                       T = T - 1;
                       GOTO OL;
                    END;
                    IF CONT_B ^= 0 THEN
                       CALL VERTICA_BANCO;
                    NUMERO_B(T) = T;
                    NOMBRE_B(T) = NOMBRE1;
                    T = T + 1;
                    CALL GOTOXY(14,T);
                    READ FILE(IN) INTO(NOMBRE2);
                    DIR_B(T) = NOMBRE2;
                    T = T + 5;
                    OL:
                    CONT_B = 1;
                    CONT = CONT + 1;
                    OOL:
                 END;
                 IF T ^= 0 THEN DO;
                    CALL PIDE_VACIO_B2;
                    BLA = 0;
                    BLS = 0;
                    DIR = KLAV2;
                 WRITE FILE(BANCOS) FROM(INFBAN) KEYFROM(DIR);
                 END;
              END;
              FINAL:
            END;
            ELSE DO;
              IF BL_S ^= 0 THEN DO;
                 DO WHILE(BL_S ^= 0);
                    DIRECCION = BL_S;
                    READ FILE(MONEDAS) INTO(INFMON) KEY(DIRECCION);
                 END;
                 PUT FILE(OUT) PAGE;
                 CALL GOTOXY(6,0);
PUT FILE(OUT) EDIT(ASCII(29),'MONEDA','SIMBOLO','SITUACION',ASCII(28))(A,A(6),X(
8),A(7),X(4),A(9),A);
                 IF CONT_M ^= 0 THEN DO;
                   DO K = 1 TO CONT_M;
PUT FILE(OUT) SKIP EDIT(MONE(K),SIMBOLO(K),
                    SITUACION_M(K))(X(6),A(10),X(8),A(6),X(4),A(1));
                    END;
                 END;
               END;
                 IF CONT_M ^= 12 THEN DO;
                    CONT_M1 = CONT_M;
                    DO WHILE(REPITA = '1'B & CONT_M1 < 12);
            RA:        CALL GOTOXY(6,CONT_M1+1);
                       READ FILE(IN) INTO(DATO);
```

```
                        CALL GOTOXY(24,CONT_M1 + 1);
                        READ FILE(IN) INTO(DATO1);
                        IF DATO = "" THEN DO;
                           REPITA = "0"B;
                           GOTO AA;
                        END;
                        CONT_M = CONT_M1;
                        IF CONT_M1 ^= 0 THEN
                           CALL VERIFICA_MONEDA;
                        CONT_M1 = CONT_M1 + 1;
                        MONE(CONT_M1) = DATO;
                        SIMBOLO(CONT_M1) = DATO1;
                        AA:
                     END;
                        CONT_M = CONT_M1;
                        WRITE FILE(MONEDAS) FROM(INFMON) KEYFROM(DIRECCION);
                  END;
                  ELSE DO;
                     PUT FILE(OUT) PAGE;
                     CALL GOTOXY(6,1);
PUT FILE(OUT) EDIT(ASCII(29),"MONEDA","SIMBOLO","SITUACION",ASCII(28))(A,A(6),X(
8),A(7),X(4),A(9),A);
            RE:      CALL GOTOXY(6,2);
                     READ FILE(IN) INTO(DATO);
                     CALL GOTOXY(24,2);
                     READ FILE(IN) INTO(DATO1);
                     IF DATO = "" THEN GOTO AC;
                     ELSE DO;
                        CALL VERIFICA_MONEDA;
                        CALL PIDE_VACIO_B1;
                        BL_S = BLOV1;
                        REWRITE FILE(MONEDAS) FROM(INFMON) KEY(DIRECCION);
                        BL_A = DIRECCION;
                        BL_S = 0;
                        DIRECCION = BLOV1;
                        CONT_M = 1;
                        MONE(CONT_M) = DATO;
                        SIMBOLO(CONT_M) = DATO1;
                        IF CONT_M = 1 THEN
               WRITE FILE(MONEDAS) FROM(INFMON) KEYFROM(DIRECCION);
                        DO WHILE(CONT_M < 12 & REPITA = "1"B);
                        KI: CALL GOTOXY(6,CONT_M+2);
                            READ FILE(IN) INTO(DATO);
                            CALL GOTOXY(24,CONT_M+2);
                            READ FILE(IN) INTO(DATO1);
                            IF DATO = "" THEN DO;
                               REPITA = "0"B;
                               GOTO AB;
                            END;
                            CALL VERIFICA_MONEDA;
                            CONT_M = CONT_M + 1;
                            MONE(CONT_M) = DATO;
                            SIMBOLO(CONT_M)= DATO1;
                        AB:
                        END;
                        WRITE FILE(MONEDAS) FROM(INFMON) KEYFROM(DIRECCION);
                     END;
                     AC:
                  END;
            END;

         END; /* CREAR    */
```

```
   MODIFICAR: PROCEDURE;

/* RUTINA QUE SE UTILIZA PARA MODIFICAR       */

      MOD = "0"B;
      IF BANDERA = "1"B THEN DO;
         IF CONT_B ^= 0 THEN DO;
            CONT_B1 = CONT_B;
            IF SITUACION(1) = "E" THEN GOTO A;
            CALL GOTOXY(14,3);
            READ FILE(IN) INTO(NOMBRE1);
            IF NOMBRE1 = "" THEN GOTO A;
            MOD = "1"B;
            NOMBRE_B(1) = NOMBRE1;
            CALL GOTOXY(14,4);
            READ FILE(IN) INTO(NOMBRE1);
            IF NOMBRE1 ^= "" THEN
               DIR_B(1) = NOMBRE1;
      A:    CONT_B1 = CONT_B1 -1;
            IF CONT_B1 ^= 0 THEN DO;
               IF SITUACION(2) = "E" THEN GOTO B;
               CALL GOTOXY(14,9);
               READ FILE(IN) INTO(NOMBRE1);
               IF NOMBRE1 = "" THEN GOTO B;
               IF MOD = "0"B THEN MOD = "1"B;
               NOMBRE_B(2) = NOMBRE1;
               CALL GOTOXY(14,10);
               READ FILE(IN) INTO(NOMBRE1);
               IF NOMBRE1 ^= "" THEN
                  DIR_B(2) = NOMBRE2;
      B:       CONT_B1 = CONT_B1 - 1;
               IF CONT_B1 ^= 0 THEN DO;
                  IF SITUACION(3) = "E" THEN GOTO C;
                  CALL GOTOXY(14,15);
                  READ FILE(IN) INTO(NOMBRE1);
                  IF NOMBRE1 = "" THEN GOTO C;
                  IF MOD = "0"B THEN MOD = "1"B;
                  NOMBRE_B(3) = NOMBRE1;
                  CALL GOTOXY(14,16);
                  READ FILE(IN) INTO(NOMBRE1);
                  IF NOMBRE1 ^= "" THEN
                     DIR_B(3) = NOMBRE1;
                  C:
               END;
            END;
            IF MOD = "1"B THEN
               REWRITE FILE(BANCOS) FROM(INFBAN) KEY(DIR);
         END;
      END;
      ELSE DO;
         IF CONT_M ^= 0 THEN DO;
            I = 0;
            DO WHILE(I < CONT_M);
               IF SITUACION_M(I+1) = "E" THEN
                  GOTO MA;
               CALL GOTOXY(6,I+1);
               READ FILE(IN) INTO(DATO);
               CALL GOTOXY(24,I+1);
               READ FILE(IN) INTO(DATO1);
               IF DATO ^= "" THEN
                  MONE(I+1) = DATO;
               IF DATO1 ^= "" THEN
                  SIMBOLO(I+1) = DATO1;
```

```
                MA;
                I = I + 1;
             END;
              REWRITE FILE(MONEDAS) FROM(INFMON) KEY(DIRECCION);
        END;
      END;

  END;  /*     MODIFICAR          */

            PIDE_VACIO_B2: PROCEDURE;

/*  RUTINA QUE SUMINISTRA LOS BLOQUES VACIOS    */

  READ FILE(BANCOS) INTO(CONTROL2) KEY(0);
     MAX_DIR2 = MAX_DIR2 + 1;
     BLOQ2 = MAX_DIR2;
  WRITE FILE(BANCOS) FROM(CONTROL2) KEYFROM(0);

END;  /* PIDE_VACIO_B2     */

PIDE_VACIO_B1: PROCEDURE;

READ FILE(MONEDAS) INTO(CONTROL1) KEY(0);
   MAX_DIR1 = MAX_DIR1 + 1;
   BLOQ1 = MAX_DIR1;
WRITE FILE(MONEDAS) FROM(CONTROL1) KEYFROM(0);

END; /*     PIDE_VACIO_B1      */

ELIMINAR: PROCEDURE;

/* RUTINA QUE ELIMINA BANCOS Y MONEDAS */

   EL = "0"B;
   IF BANDERA = "1"B THEN DO;
      IF CONT_B ^= 0 THEN DO;
         CONT_B1 = CONT_B;
         IF SITUACION(1) = "E" THEN GOTO D;
E:       CALL GOTOXY(16,1);
         READ FILE(IN) INTO(NOMBRE10);
         IF NOMBRE10 = "" THEN GOTO D;
         IF NOMBRE10 ^= "E" THEN DO;
            CALL ERROR1;
            GOTO E;
         END;
         SITUACION(1) = "E";
         EL = "1"B;
D:       CONT_B1 = CONT_B1 - 1;
         IF CONT_B1 ^= 0 THEN DO;
            IF SITUACION(2) = "E" THEN GOTO F;
G:          CALL GOTOXY(16,7);
            READ FILE(IN) INTO(NOMBRE10);
            IF NOMBRE10 = "" THEN GOTO F;
            IF NOMBRE10 ^= "E" THEN DO;
               CALL ERROR1;
               GOTO G;
            END;
            SITUACION(2) = "E";
            IF EL = "0"B THEN EL = "1"B;
F:          CONT_B1 = CONT_B1 - 1;
            IF CONT_B1 ^= 0 THEN DO;
               IF SITUACION(3) = "E" THEN GOTO K;
Z:             CALL GOTOXY(16,13);
```

```
                READ FILE(IN) INTO(NOMBRE10);
                IF NOMBRE10 = "" THEN GOTO K;
                IF NOMBRE10 ^= "E" THEN DO;
                   CALL ERROR1;
                   GOTO Z;
                END;
                SITUACION(3) = "E";
                IF EL  = "0"B THEN EL = "1"B;
K:
             END;
          END;
          IF EL = '1"B THEN
             REWRITE FILE(BANCOS) FROM(INFBAN) KEY(DIR);
       END;
END;
ELSE DO;
       IF CONT_M ^= 0 THEN DO;
          I = 0;
          DO WHILE(I < CONT_M);
             IF SITUACION_M(I+1) = "E" THEN
                GOTO ZZ;
             CALL GOTOXY(34,I+1);
             READ FILE(IN) INTO(NOMBRE10);
             IF (NOMBRE10 = "" | NOMBRE10 ^= "E") THEN
                GOTO ZZ;
             SITUACION_M(I+1) = "E";
             ZZ:
             I = I + 1;
          END;
             WRITE FILE(MONEDAS) FROM(INFMON) KEYFROM(DIRECCION);
       END;
   END;

END; /*     ELIMINAR       */

ERROR1: PROCEDURE;

     CALL GOTOXY(0,23);
          PUT FILE(OUT) EDIT('SI VA A ELIMINAR DEBE COLOCAR UNA E')(A);
END;  /* ERROR1  */

/* RUTINA QUE LEE EL BLOQUE SIGUIENTE    */

ADELANTE: PROCEDURE;

    IF BANDERA = "1"B THEN DO;
       IF BLS ^= 0 THEN DO;
          DIR = BLS;
          READ FILE(BANCOS) INTO(INFBAN) KEY(DIR);
          GOTO ADE;
       END;
       ELSE DO;
          CALL GOTOXY(0,23);
          PUT FILE(OUT) EDIT('NO EXISTE MAS PAGINA')(A);
          GOTO REPETIR;
       END;
    END;
    ELSE DO;
       IF BL_S ^= 0 THEN DO;
          DIRECCION = BL_S;
          READ FILE(MONEDAS) INTO(INFMON) KEY(DIRECCION);
          GOTO ADE1;
      END;
```

```
      ELSE DO;
         CALL GOTOXY(0,23);
         PUT FILE(OUT) EDIT("NO EXISTE MAS PAGINA")(A);
         GOTO REPETIR;
      END;
   END;

END; /* ADELANTE    */

/*  RUTINA QUE RETROCEDE UN BLOQUE   */

RETROCEDE: PROCEDURE;

  IF BANDERA = '1'B THEN DO;
     IF BLA ^= 0 THEN DO;
        DIR = BLA;
        READ FILE(BANCOS) INTO(INFBAN) KEY(DIR);
        GOTO ADE;
     END;
     ELSE DO;
        CALL GOTOXY(0,23);
        PUT FILE(OUT) EDIT("NO EXISTE BLOQUE ANTERIOR")(A);
        GOTO REPETIR;
     END;
  END;
  ELSE DO;
     IF BL_A ^= 0 THEN DO;
        DIRECCION = BL_A;
        READ FILE(MONEDAS) INTO(INFMON) KEY(DIRECCION);
        GOTO ADE1;
     END;
     ELSE DO;
        CALL GOTOXY(0,23);
        PUT FILE(OUT) EDIT("NO EXISTE PAGINA ANTERIR")(A);
        GOTO REPETIR;
     END;
  END;

END; /*   RETROCEDE    */

IMPRIMIR: PROCEDURE;

DCL       GUAR_DIR                    BIN FIXED;

OPEN FILE(IMP) TITLE('@LPT') STREAM OUTPUT PRINT;

IF BANDERA = '1'B THEN DO;
   GUAR_DIR = DIR;
   DIR = 1;
   READ FILE(BANCOS) INTO(INFBAN) KEY(DIR);
   IF CONT_B = 0 THEN DO;
      CALL GOTOXY(0,23);
      PUT FILE(OUT) EDIT("NO HAY BANCOS QUE IMPRIMIR")(A);
   END;
   ELSE DO;
      DO I = 1 TO CONT_B;
         PUT FILE(IMP) SKIP EDIT('SITUACION:',SITUACION(I),'NOMBRE:',NOMBRE_B(I
,'DIR:',DIR_B(I))(A,A,SKIP,A,A,SKIP,A,A);
      PUT FILE(IMP) SKIP(2);
      END;
   END;
   DO WHILE(BLS ^= 0);
      DIR = BLS;
      READ FILE(BANCOS) INTO(INFBAN) KEY(DIR);
```

```
CALL GOTOXY(14,T);
```

```
DIRECCION1 = DIRECCION;
   J = 0;
   DO WHILE(J < CONT_M & ENCONTRADO = "0"B);
      IF DATO = MONE(J+1) THEN DO;
         ENCONTRADO = "1"B;
         SITUACION_M(J+1) = "";
         WRITE FILE(MONEDAS) FROM(INFMON) KEYFROM(DIRECCION1);
         READ FILE(MONEDAS) INTO(INFMON) KEY(DIRECCION1);
         DIRECCION = DIRECCION1;
         PUT FILE(OUT) PAGE;
         CALL GOTOXY(6,0);
PUT FILE(OUT) EDIT("MONEDA","SIMBOLO","SITUACION")(A(6),X(8),A(7),X(4),A(9));
         IF CONT_M ^= 0 THEN DO;
            DO K = 1 TO CONT_M;
PUT FILE(OUT) SKIP EDIT(MONE(K),SIMBOLO(K),
SITUACION_M(K))(X(6),A(10),X(8),A(6),X(4),A(1));
            END;
         END;
         GOTO REPETIR;
      END;
      J = J + 1;
   END;
   IF ENCONTRADO = "0"B THEN DO;
      DO WHILE(BL_A ^= 0);
         DIRECCION = BL_A;

      DO I = 1 TO CONT_B;
         PUT FILE(IMP) SKIP EDIT("SITUACION:",SITUACION(I),"NOMBRE:",NOMBRE_B(I)
,"DIR:",DIR_B(I))(A,A,SKIP,A,A,SKIP,A,A);
      PUT FILE(IMP) SKIP(2);
      END;
   END;
   DIR = GUAR_DIR;
 END;
 ELSE DO;
   GUAR_DIR = DIRECCION;
   READ FILE(MONEDAS) INTO(INFMON) KEY(DIRECCION);
   IF CONT_M = 0 THEN DO;
      CALL GOTOXY(0,23);
      PUT FILE(OUT) EDIT("NO HAY MONEDAS QUE IMPRIMIR")(A);
   END;
   ELSE DO;
PUT FILE(IMP) SKIP EDIT("MONEDA","SIMBOLO","SITUACION")(A(6),X(8),A(7),X(4),A(9)
);
      DO I = 1 TO CONT_M;
         PUT FILE(IMP) SKIP EDIT(MONE(I),SIMBOLO(I),SITUACION_M(I))(X(6),A(10),X
(8),A(6),X(4),A(1));
      END;
   END;
   DO WHILE(BL_S ^= 0);
      DIRECCION = BL_S;
      READ FILE(MONEDAS) INTO(INFMON) KEY(DIRECCION);
      DO I = 1 TO CONT_M;
PUT FILE(IMP) SKIP EDIT(MONE(I),SIMBOLO(I),SITUACION_M(I))(X(6),A(10),X(8),A(6),
X(4),A(1));
      END;
   END;
   DIRECCION = GUAR_DIR;

END;

 CLOSE FILE(IMP);

END; /*     IMPRIMIR     */
```

```
CUADRE_ACTIVIDAD: PROCEDURE(CONT_AU_ACEP,VECTOR_GUAR_NUM,
CADENA_LEIDA,DIRE_PANTA,NUMERO_ACTIVIDAD,CUADRE);

/*****************************************************************************
 *   PROGRAMA QUE REALIZA CUADRE DE MONTOS EN LA FACTURA                     *
 *   (LA SUMA DE LOS MONTOS PARCIALES DE LOS LIBROS                          *
 *    DEBE SER IGUAL AL MONTO TOTAL PARA QUE ALLA                            *
 *    CUADRE DE MONTOS)                                                      *
 *                                                                           *
 *   PARAMETROS:                                                             *
 *      CONT_AU_ACEP: VARIABLE QUE CONTIENE LA CANTIDAD DE                   *
 *      AUTORES Y/O PP ANEXADAS AL PROCESO QUE SE ESTA CREANDO               *
 *      VECTOR_GUAR_NUM: VECTOR QUE GUARDA LOS NUMEROS                       *
 *      CORRESPONDIENTES DE LOS AUTORES Y/O PP A SER                         *
 *      ANEXADOS AL PROCESO                                                  *
 *      CADENA_LEIDA: VARIABLE QUE CONTIENE LA INFORMACION                   *
 *      PRINCIPAL AL PROCESO QUE SE ESTA CREANDO                             *
 *      DIRE_PANTA:DIRECCION DEL REGISTRO CORRESPONDIENTE A                  *
 *      LA PANTALLA DEL PROCESO                                              *
 *      NUMERO_ACTIVIDAD: NUMERO DE LA ACTIVIDAD A SER CREADA                *
 *      CUADRE: VARIABLE QUE INDICA SI EXISTE O NO CUADRE                    *
 *            0: NO CUADRAN LOS MONTOS                                       *
 *            1: SI CUADRAN LOS MONTOS                                       *
 *                                                                           *
 *   VARIABLES:                                                              *
 *      NUMERO_ARC: NUMERO DEL ARCHIVO DEL PROCESO A SER CREADO              *
 *      NUMERO_INDICE: NUMERO DEL INDICE CORRESPONDIENTE AL                  *
 *      PROCESO                                                              *
 *      CONT_PAR_BS : VARIABLE QUE CONTIENE LA SUMA DE CADA UNO              *
 *      DE MONTOS DE LOS LIBROS Y/O REVISTAS                                 *
 *      MONTO_BS: VARIABLE QUE CONTIENE LA CANTIDAD TOTAL EN BS              *
 *      FLETE_TOT: MONTO TOTAL DE FLETE                                      *
 *      CONT_PAR_FLETE: MONTO DEL FLETE POR AUTOR/PP                         *
 *      CONT_AU: NUMERO DE AUTORES/PP ANEXADOS A LA FACTURA                  *
 *                                                                           *
 *   LLAMA A:                                                                *
 *      BUSCA                                                                *
 *      GOTOXY                                                               *
 *                                                                           *
 *   LLAMADA POR:                                                            *
 *      MUESTRA_INFORMACION                                                  *
 *                                                                           *
 *   ARCHIVOS:                                                               *
 *      CONTROL_ARCHIVOS: CONTIENE INFORMACION DE LOS PRINCIPALES            *
 *      ARCHIVOS DEL SISTEMA                                                 *
 *                                                                           *
 *   MACRO ALGORITMO DEL PROGRAMA:                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
```

```
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 *                                                                                                  *
 ****************************************************************************************************/

DCL
CK                                                          CHAR(1) VAR,
(FLETE_TOT,FLETE_PAR,CONT_PAR_FLETE)                        DEC FIXED(10,2),
(CONT_AU_ACEP,DIRE_PANTA,I,POS2)                            BIN FIXED,
(MONTO_BS,CONT_PAR_BS)                                      DEC FIXED(10,2),
 CADENA_LETDA                                               CHAR(1000) VAR,
 CUADRE                                                     BIT(1),
(IN,OUT,INDICE,ARCH_DATOS,CONTROL_ARCHIVO)                  FILE,
01 BLQ_MONO PP,
   02 CADENA_LETDA2                                         CHAR(1000) VAR,
TESAURO                                                     FILE VARIABLE,
(NUMERO_ARC,NUMERO_INDICE,LONGITUD,POS1,LON1)                    BIN FIXED,
RA_ENC                                                           BIT(1),
(POS_ABS,PF_CM,LON_CM,PF_CB,ULT_N,POS_RE,VIA(5))                 BIN FIXED,
(RES_CM,CLAVE)                                              CHAR(120) VAR,
01 INF,
   02 APU_RAIZ                                                   BIN FIXED,
   02 NO_NTV                                                     BIN FIXED,
   02 NO_NODO                                                    BIN FIXED,
   02 APU_PROX_VACIO                                             BIN FIXED,
01 NODO,
   02 CAB_NODO,
      03 NUM_N                                                   BIN FIXED,
      03 NTV                                                     BIN FIXED,
      03 NC                                                      BIN FIXED,
      03 IN                                                      BIN FIXED,
      03 VECINA_I                                                BIN FIXED,
      03 VECINA_D                                                BIN FIXED,
   02 ENTRADA                                                    CHAR(1009) VAR,
01 CONTROL_ARCH(14),
   02 TIPO                                                       CHAR(1),
   02 N_ARC                                                      BIN FIXED,
   02 LON_REG                                                    BIN FIXED,
   02 ULT_VACIO                                                  BIN FIXED,
   02 MAX_DIR                                                    BIN FIXED,
   02 NOMBRE_ARC                                                 CHAR(20) VAR,
(NUMERO_ACTIVIDAD,NUMERO_ACTIVIDAD_1)                       CHAR(12) VAR,
VECTOR_GUAR_NUM(30)                                         BIN FIXED,
DIR                                                         DEC FIXED;
DCL
BUSCA_ENTRY(FILE VARIABLE,CHAR(120) VAR,BIT(1),BIN FIXED,CHAR(120) VAR,
BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,BIN FIXED,
(5) BIN FIXED,BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,2 BIN FIXED,
2 BIN FIXED),
GOTOXY_ENTRY(BIN FIXED,BIN FIXED);

CK = ASCII(11);
TESAURO = INDICE;
NUMERO_ARC = 1;
NUMERO_INDICE = 11;
```

```
CUADRE = "0"B;
CONT_PAR_BS = 0;
CONT_PAR_FLETE = 0;

READ FILE(CONTROL_ARCHIVO) INTO(CONTROL_ARCH) KEY(1);
I = 1;
DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_ARC);
   I = I + 1;
END;
LONGITUD = CONTROL_ARCH(I).LON_REG;
CLOSE FILE(ARCH_DATOS);
OPEN FILE(ARCH_DATOS) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
DIRECT KEYED ENV(RECSIZE(LONGITUD));
CLOSE FILE(INDICE);
I = 1;
DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
   I = I + 1;
END;
OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
DIRECT KEYED ENV(RECSIZE(1024));
   NUMERO_ACTIVIDAD_1 = "FA"||NUMERO_ACTIVIDAD;
   POS1 = 1;
   LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
   POS1 = POS1 + LON1;
   LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
   POS1 = POS1 + LON1;
   LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
   POS1 = POS1 + LON1;
   LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
   MONTO_BS = DECIMAL(SUBSTR(CADENA_LEIDA,POS1+1,LON1-1),10,2);
   POS1 = POS1 + LON1;
   LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
   FLETE_TOT= DECIMAL(SUBSTR(CADENA_LEIDA,POS1+1,LON1-1),10,2);
   DO I = 1 TO CONT_AU_ACEP;
      CLAVE = CHAR(VECTOR_GUAR_NUM(I));
   CALL BUSCA(TESAURO,CLAVE,RA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
   PF_CB,VIA,UBT_N,POS_RE,INF);
       DTR = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+2,1)))*10000 +
       RANK(SUBSTR(ENTRADA,POS_ABS+3,1))*100 +
       RANK(SUBSTR(ENTRADA,POS_ABS+4,1));
       READ FILE(ARCH_DATOS) INTO(BLQ_MONO_PP) KEY(DIR);
       POS1 = 1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
  DO WHILE(NUMERO_ACTIVIDAD_1 ^= SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1));
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
  END;
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
CONT_PAR_BS=CONT_PAR_BS+DECIMAL(SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1),10,2);
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
FLETE_PAR = DECIMAL(SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1),10,2);
CONT_PAR_FLETE=CONT_PAR_FLETE+FLETE_PAR;
END;
IF MONTO_BS = CONT_PAR_BS THEN
   CUADRE = "1"B;
IF CUADRE = "1"B THEN DO;
   IF CONT_PAR_FLETE ^= FLETE_TOT THEN DO;
      CONT_PAR_FLETE = FLETE_TOT / CONT_AU_ACEP;
      DO I = 1 TO CONT_AU_ACEP;
         CLAVE = CHAR(VECTOR_GUAR_NUM(I));
   CALL BUSCA(TESAURO,CLAVE,RA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
```

```
   PF_CR,VTA,ULT_N,POS_RF,INF);
       DIR = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+2,1)))*10000 +
       RANK(SUBSTR(ENTRADA,POS_ABS+3,1))*100 +
       RANK(SUBSTR(ENTRADA,POS_ABS+4,1));
       READ FILE(ARCH_DATOS) INTO(BLQ_MONO_PP) KEY(DIR);
       POS1 = 1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
  DO WHILE(NUMERO_ACTIVIDAD_I ^= SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1));
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
  END;
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
       POS2 = POS1 + LON1;
       CADENA_LEIDA2 = SUBSTR(CADENA_LEIDA2,1,POS1-1)||
       ASCII(LENGTH(CHAR(CONT_PAR_FLETE))+1) || CHAR(CONT_PAR_FLETE) ||
       SUBSTR(CADENA_LEIDA2,POS2);
       WRITE FILE(ARCH_DATOS) FROM(BLQ_MONO_PP) KEYFROM(DIR);
    END;
 END;
END;
END;      /*   CUADRE_ACTIVIDAD   */
```

```
INSERTA: PROCEDURE(TESAURO,RES_CB,API_INV1,NREF,API_INV2);

/*****************************************************************************
*                                                                            *
*      INSERTA                                                               *
*                                                                            *
*     RUTINA QUE INSERTA UNA RES_CB EN EL ARBOL                              *
*                                                                            *
*      RUTINAS QUE LLAMA:  BUSCA Y ROMPER_NODO                               *
*                                                                            *
*      LLAMADA POR:                                                          *
*                                                                            *
*      PARAMETROS:                                                           *
*                                                                            *
*        DE ENTRADA:                                                         *
*          RES_CB: RES_CB QUE SE VA A INSERTAR                               *
*          AP_INV: APUNTADOR ASOCIADO A LA RES_CB                            *
*          NREF: NUMERO DE REFERENCIAS DE LA RES_CB                          *
*                                                                            *
*****************************************************************************/

/*
                DECLARACION DE VARIABLES
                                                                        */
DCL
    (APN1,APNN1,U1,T_N)             BIN FIXED,
    (N1,N2)                         DEC FIXED(5),
    NREF                            BIN FIXED,
    NREF1                           BIN FIXED,
    LON_CB                          BIN FIXED,
    LON_CM                          BIN FIXED,
    T                               BIN FIXED,
    LON_REG_PAR                     BIN FIXED,
    AP                              DEC FIXED,
    PF_CB                           BIN FIXED,
    PF_CM                           BIN FIXED,
    TMAXN                           BIN FIXED,
    POS_ABS                         BIN FIXED,
    POS_RF                          BIN FIXED,
    (API_INV1,API_INV2)             DEC FIXED,
    B1                              BIT (20),
    B2                              BIT (20),
    BA_FIN                          BIT(1),
    BA_ENC                          BIT(1),
    BA_DEC                          BIT(1),
    IGUAL                           BIT(1),
    APU                             CHAR(8) VAR STATIC INIT(""),
    RES_CM                          CHAR (120)VAR,
    RES_CB                          CHAR (120)VAR, /* RES_CB = CB A PARTIR DE PFCB +
1   */
    VTA(5)                          BIN FIXED,
1   CNF,
    2 APU_RAIZ                      BIN FIXED,
    2 NO_NIV                        BIN FIXED,
    2 NO_NODO                       BIN FIXED,
    2 APU_PROX_VACIO                BIN FIXED,
     1 NODO,
       2 CAB_NODO,
          3 NUM_N                   BIN FIXED,
          3 NIV                     BIN FIXED,
          3 NC                      BIN FIXED,
          3 TN                      BIN FIXED,
          3 VECINA_I                BIN FIXED,
          3 VECINA_D                BIN FIXED,
```

```
     2 ENTRADA                       CHAR(1009) VAR,
       TESAURO                       FILE VARIABLE;
DCL
   BUSCA          ENTRY(FILE VARIABLE,CHAR(120)VAR,BIT(1),BIN FIXED,CHAR(120)VAR,
                  BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,
                  3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
                  2 CHAR(1009) VAR,BIN FIXED,(5) BIN FIXED,BIN FIXED,
                  BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,2 BIN FIXED,
                  2 BIN FIXED),
 ROMPE_NODO      ENTRY(FILE VARIABLE,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
                  3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,
                  (5) BIN FIXED,CHAR(120)VAR,BIN FIXED,BIN FIXED,BIN FIXED,
                  BIT(1),BIN FIXED,BIN FIXED);
/*
                   PROGRAMA PRINCIPAL
                                                                    */
N1 = API_INV1;
N2 = API_INV2;
LON_REG_PAR = 10;
BA_FIN = '0'B;
TMAXN = 879; /*  1009 - (120+ 10) = 879  */
LON_CB = LENGTH(RES_CB);
CALL BUSCA (TESAURO,RES_CB,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,PF_CB,VIA,UL
 N,POS_RE,ONF);

/*   ACTUALIZA EL PRELON_REG_PAR, LONG. Y RESTO DE LA RES_CB EN MANO  */
DO WHILE (^BA_FIN);
   I = 0;
   IF PF_CB = PF_CM THEN DO;
      CALL ACT_PF;
      PF_CM = PF_CM + I;
      LON_CM = LON_CM - I;
   END;

/*  INSERTA LA NUEVA RES_CB, ACTUALIZA APUNTADORES Y RESCRIBE LA
       PANTNA                                                       */

   IF ^BA_ENC THEN DO;
      IF LON_REG_PAR = 10 THEN DO;
         B1 = SUBSTR(UNSPEC(N1),1,20);
         B2 = SUBSTR(UNSPEC(N2),1,20);
         APU = ASCII(BIN(SUBSTR(B1,1,4)))||ASCII(BIN(SUBSTR(B1,5,4))*10+
               BIN(SUBSTR(B1,9,4)))||ASCII(BIN(SUBSTR(B1,13,4))*10+
               BIN(SUBSTR(B1,17)))||ASCII(NREF/256)||ASCII(NREF)||
               ASCII(BIN(SUBSTR(B2,1,4)))||ASCII(BIN(SUBSTR(B2,5,4))*10+
               BIN(SUBSTR(B2,9,4)))||ASCII(BIN(SUBSTR(B2,13,4))*10+
               BIN(SUBSTR(B2,17)));
      END;
      ELSE
         APU = ASCII(APNT/256)||ASCII(APNT);
      IF POS_ABS > 1 THEN DO;
         ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1)||ASCII(PF_CB)||
          ASCII(LON_CB-PF_CB+LON_REG_PAR)||APU||SUBSTR(RES_CB,PF_CB+1)||
          ASCII(PF_CM)||ASCII(LON_CM)||SUBSTR(ENTRADA,POS_ABS+2,
          LON_REG_PAR-2)||SUBSTR(ENTRADA,POS_ABS+LON_REG_PAR+1);
         TN = TN + LON_CB - PF_CB + LON_REG_PAR - 1;
      END;
      ELSE DO;
         ENTRADA = ASCII(0)||ASCII(LON_CB+LON_REG_PAR)||APU||RES_CB||
          ASCII(PF_CM)||ASCII(LON_CM)||SUBSTR(ENTRADA,POS_ABS+2,
          LON_REG_PAR-2)||SUBSTR(ENTRADA,POS_ABS+LON_REG_PAR+1);
         TN = TN + LON_CB + LON_REG_PAR - 1;
      END;
      IF LON_REG_PAR = 4 THEN DO;
```

```
         LON_CM = RANK(SUBSTR(ENTRADA,POS_ABS+1,1));
         POS_ABS = POS_ABS + LON_CM;
         SUBSTR(ENTRADA,POS_ABS+2,2) = ASCII(APNNU/256)||ASCII(APNNU);
      END;
      BA_FIN = "1"B;
      NC = NC + 1;
      IF TN > TMAXN THEN CALL ROMPE_NODO(TESAURO,NODO,VIA,RES_CB,LON_CB,
                APNI,APNNU,BA_FIN,LON_REG_PAR,ULT_N);
      ELSE REWRITE FILE (TESAURO) FROM (NODO) KEY (NUM_N);
      IF ^BA_FIN THEN CALL BUSCA_EN_NODO;
   END;
   ELSE RETURN;
END; /* FIN DEL DO WHILE */

/*************************************************************************
    RUTINA QUE BUSCA LA POSICION A INSERTAR EL SEPARADOR EN LA MADRE
*************************************************************************/

BUSCA_EN_NODO:PROCEDURE;
BA_DEC = "0"B;
POS_ABS = 1;
POS_RE = 0;
PF_CB = 0;
LON_CM = 0;

/*  LAZO DE REPETICION DE BUSQUEDA  */

REPITA:
   POS_RE = POS_RE + 1;
   POS_ABS = POS_ABS + LON_CM;
   IF POS_RE = NC+1 THEN RETURN;
   PF_CM = RANK(SUBSTR(ENTRADA,POS_ABS,1));
   LON_CM = RANK(SUBSTR(ENTRADA,POS_ABS+1,1));
   IF PF_CM = PF_CB THEN DO;
      RES_CM = SUBSTR(ENTRADA,POS_ABS+4,LON_CM-4);
      IF SUBSTR(RES_CB,PF_CB+1) > RES_CM THEN DO;
         CALL ACT_PF;
         PF_CB = PF_CB + 1;
      END;
/*  SI CB <= CM  ->  SE ENCONTRO LA POSICION  */

      ELSE
         BA_DEC = "1"B;
   END;
   ELSE
      IF PF_CB > PF_CM THEN
         BA_DEC = "1"B;
IF  ^(BA_DEC) THEN GOTO REPITA;
RETURN;
END; /*  FIN DE BUSCA_EN_NODO  */

ACT_PF: PROCEDURE;
   I = 0;
   S:  I = I + 1;
       IGUAL = "0"B;
       IF SUBSTR(RES_CB,PF_CB+1,1) = SUBSTR(RES_CM,I,1) THEN DO;
          IGUAL = "1"B;
          IF I < LON_CM-LON_REG_PAR & I+PF_CB < LON_CB  THEN GOTO S;
       END;
       IF ^IGUAL  THEN I = I - 1;
RETURN;
END; /*  FIN DE ACT_PF  */
END;
```

```
REALIZA_CARTA: PROCEDURE(DIRE_PANTA,CONTADOR_NOTA,VECTOR_NOTA,
NUMERO_ACTIVIDAD);

/*************************************************************************
 *                                                                       *
 *  PROGAMA ENCARGADO DE REALIZAR LA INFORMACION NECESARIA               *
 *  PARA LA REALIZACION DE LAS CARTAS                                    *
 *                                                                       *
 *    PARAMATROS:                                                        *
 *       DIRE_PANTA: DIRECCION DEL REGISTRO CORRESPONDIENTE              *
 *       AL PROCESO QUE SE ESTA CREANDO                                  *
 *       CONTADOR_NOTAS: VARIABLE QUE CONTIENE LA CANTIDAD               *
 *       DE LINEAS CON NOTAS EN EL VECTOR DE NOTAS                       *
 *       VECTOR_NOTA: VECTOR DONDE SE ENCUENTRAN LAS NOTAS               *
 *       QUE VAN A SER ANEXADAS A LA CARTA                               *
 *       NUMERO_ACTIVIDAD: NUMERO DEL PROCESO QUE SE                     *
 *       ESTA CREANDO                                                    *
 *                                                                       *
 *    VARIABLES:                                                         *
 *       NUMERO_INDICE: CONTIENE EL NUMERO DEL ARCHIVO                   *
 *       INDICE DEL PROCESO                                              *
 *       NUMERO_ARC: CONTIENE EL NUMERO DEL ARCHIVO DE                   *
 *       INFORMACION DONDE SERA GUARDADO EL PROCESO                      *
 *                                                                       *
 *    LLAMADA POR:                                                       *
 *       MUESTRA_INFORMACION                                             *
 *                                                                       *
 *    LLAMA A:                                                           *
 *       BUSCA                                                           *
 *      GOTOXY                                                           *
 *                                                                       *
 *    ARCHIVOS:                                                          *
 *       PEDIDO_TMP: GUARDA LA INFORMACION DEL PEDIDO                    *
 *       PEDIDO_PROF:GUARDA LA INFORMACION DE LA PROFORMA                *
 *       PEDIDO_CHEQ:GUARDA LA INFORMACION DE LA SOLICITUD               *
 *       DE CHEQUE                                                       *
 *       PEDIDO_PAG: GUARDA LA INFORMACION DE PAGO                       *
 *                                                                       *
 *    MACRO ALGORITMO DEL PROGRAMA:                                      *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *************************************************************************/

DCL
(RES_CM,CLAVE,CLAVE2,AUTOR)                    CHAR(120) VAR,
```

```
(AP2,DIR)                                   DEC FIXED,
(NUM_AUI,CANT_AU,LONGITUD,LON,POS)          BIN FIXED,
(DIRE_PANTA,CONTADOR_NOTA,J,K,I)            BIN FIXED,
VECTOR_NOTA(15)                             CHAR(70) VAR,
NUMERO_ACTIVIDAD                            CHAR(12) VAR,
(PEDIDO_IMP,PEDIDO_CHEQ,
PEDIDO_PAG,CONTROL_ARCHIVO)                 FILE,
(OUT,IN,INDICE,ARCH_DATOS)                  FILE,
PEDIDO_PROF                                 FILE,
(NUMERO_ARC,NUMERO_INDICE)                                  BIN FIXED,
01 BLQ_MONO_PF,
   02 CADENA_LEIDA2                                         CHAR(1000) VAR,
01 BLQ_PROF,
   02 CADENA_PROF                                           CHAR(1000) VAR,
01 BLQ_PED,
   02 CADENA_PED                                            CHAR(1000) VAR,
01 BLQ_PAG,
   02 CADENA_PAG                                            CHAR(1000) VAR,
01 BLQ_CHEQ,
   02 CADENA_CHEQ                                           CHAR(1000) VAR,
TESAURO                                                     FILE VARIABLE,
RA_ENC                                                      BIT(1),
(POS_ABS,PF_CM,LON_CM,PF_CB,ULT_N,POS_RE,VTA(5))            BIN FIXED,
01 TNF,
   02 APU_RAIZ                                              BIN FIXED,
   02 NO_NTU                                                BIN FIXED,
   02 NO_NODO                                               BIN FIXED,
   02 APU_PROX_VACIO                                        BIN FIXED,
01 NODO,
   02 CAB_NODO,
      03 NUM_N                                              BIN FIXED,
      03 NTU                                                BIN FIXED,
      03 NC                                                 BIN FIXED,
      03 TN                                                 BIN FIXED,
      03 VECINA_I                                           BIN FIXED,
      03 VECINA_D                                           BIN FIXED,
   02 ENTRADA                                               CHAR(1009) VAR,
01 CONTROL_ARCH(1A),
   02 TIPO                                                  CHAR(1),
   02 N_ARC                                                 BIN FIXED,
   02 LON_REG                                               BIN FIXED,
   02 ULT_VACIO                                             BIN FIXED,
   02 MAX_DIR                                               BIN FIXED,
   02 NOMBRE_ARC                                            CHAR(20) VAR;
DCL
BUSCA_ENTRY(FILE VARIABLE,CHAR(120) VAR,BIT(1),BIN FIXED,CHAR(120) VAR,
BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,BIN FIXED,
(5) BIN FIXED,BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,2 BIN FIXED,
2 BIN FIXED),
GOTOXY( ENTRY(BIN FIXED,BIN FIXED);

TESAURO = INDICE;
READ FILE(CONTROL_ARCHIVO) INTO(CONTROL_ARCH) KEY(1);

IF DIRE_PANTA = 7 THEN DO;
   NUMERO_INDICE = 16;
   NUMERO_ARC = 4;
   CLAVE = NUMERO_ACTIVIDAD;
   CALL BUSCA_INDICE;
   READ FILE(ARCH_DATOS) INTO(BLQ_CHEQ) KEY(DIR);
   OPEN FILE(PEDIDO_CHEQ) OUTPUT ENVIRONMENT(APPEND);
   PUT FILE(PEDIDO_CHEQ) SKIP EDIT('/*')(A);
```

```
    PUT FILE(PEDIDO_CHEQ) SKIP EDIT(CADENA_CHEQ)(A);
    PUT FILE(PEDIDO_CHEQ) SKIP EDIT("/")(A);
    IF CONTADOR_NOTA ^= 0 THEN DO;
       PUT FILE(PEDIDO_CHEQ) SKIP EDIT("///")(A);
       DO I = 1 TO CONTADOR_NOTA;
          PUT FILE(PEDIDO_CHEQ) SKIP EDIT(VECTOR_NOTA(I))(A);
       END;
       PUT FILE(PEDIDO_CHEQ) SKIP EDIT("///")(A);
  END;
END;
IF DIRE_PANTA = 4 THEN DO;
    NUMERO_INDICE = 13;
    NUMERO_ARC = 3;
    CLAVE = NUMERO_ACTIVIDAD;
    CALL BUSCA_INDICE;
    READ FILE(ARCH_DATOS) INTO(BLQ_PROF) KEY(DIR);
    OPEN FILE(PEDIDO_PROF) OUTPUT ENVIRONMENT(APPEND);
    PUT FILE(PEDIDO_PROF) SKIP EDIT("/")(A);
    PUT FILE(PEDIDO_PROF) SKIP EDIT(CADENA_PROF)(A);
    PUT FILE(PEDIDO_PROF) SKIP EDIT("/")(A);
    PUT FILE(PEDIDO_PROF) SKIP EDIT("//")(A);
    CALL GRABA_AUTOR;
    PUT FILE(PEDIDO_PROF) SKIP EDIT("//")(A);
    IF CONTADOR_NOTA ^= 0 THEN DO;
       PUT FILE(PEDIDO_PROF) SKIP EDIT("///")(A);
       DO I = 1 TO CONTADOR_NOTA;
          PUT FILE(PEDIDO_PROF) SKIP EDIT(VECTOR_NOTA(I))(A);
       END;
       PUT FILE(PEDIDO_PROF) SKIP EDIT("///")(A);
  END;
END;
IF DIRE_PANTA = 6 THEN DO;
    NUMERO_INDICE = 15;
    NUMERO_ARC = 5;
    CLAVE = NUMERO_ACTIVIDAD;
    CALL BUSCA_INDICE;
    READ FILE(ARCH_DATOS) INTO(BLQ_PED) KEY(DIR);
    OPEN FILE(PEDIDO_TMP) OUTPUT ENVIRONMENT(APPEND);
    IF LENGTH(CADENA_PED) > 250 THEN GOTO FF;
    PUT FILE(PEDIDO_TMP) SKIP EDIT("/")(A);
    PUT FILE(PEDIDO_TMP) SKIP EDIT(CADENA_PED)(A);
    PUT FILE(PEDIDO_TMP) SKIP EDIT("/")(A);
FF:PUT FILE(PEDIDO_TMP) SKIP EDIT("//")(A);
    CALL GRABA_AUTOR;
    PUT FILE(PEDIDO_TMP) SKIP EDIT("//")(A);
    IF CONTADOR_NOTA ^= 0 THEN DO;
       PUT FILE(PEDIDO_TMP) SKIP EDIT("///")(A);
       DO I = 1 TO CONTADOR_NOTA;
          PUT FILE(PEDIDO_TMP) SKIP EDIT(VECTOR_NOTA(I))(A);
       END;
       PUT FILE(PEDIDO_TMP) SKIP EDIT("///")(A);
  END;
END;
IF DIRE_PANTA = 8 THEN DO;
    NUMERO_INDICE = 17;
    NUMERO_ARC = 7;
    CLAVE = NUMERO_ACTIVIDAD;
    CALL BUSCA_INDICE;
    READ FILE(ARCH_DATOS) INTO(BLQ_PAG) KEY(DIR);
    OPEN FILE(PEDIDO_PAG) OUTPUT ENVIRONMENT(APPEND);
    PUT FILE(PEDIDO_PAG) SKIP EDIT("/")(A);
    PUT FILE(PEDIDO_PAG) SKIP EDIT(CADENA_PAG)(A);
    PUT FILE(PEDIDO_PAG) SKIP EDIT("/")(A);
```

```
   PUT FILE(PEDIDO_PAG) SKIP EDIT('//')(A);
   CALL GRABA_AUTOR;
   PUT FILE(PEDIDO_PAG) SKIP EDIT('//')(A);
   IF CONTADOR_NOTA ^= 0 THEN DO;
      PUT FILE(PEDIDO_PAG) SKIP EDIT('///')(A);
      DO I = 1 TO CONTADOR_NOTA;
         PUT FILE(PEDIDO_PAG) SKIP EDIT(VECTOR_NOTA(I))(A);
      END;
      PUT FILE(PEDIDO_PAG) SKIP EDIT('///')(A);
  END;
END;

BUSCA_INDICE: PROCEDURE;

I = 1;
DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
   I = I + 1;
END;
CLOSE FILE(INDICE);
OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) INPUT
DIRECT KEYED ENV(RECSIZE(1024));
I = 1;
DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_ARC);
   I = I + 1;
END;
LONGITUD = CONTROL_ARCH(I).LON_REG;
CLOSE FILE(ARCH_DATOS);
OPEN FILE(ARCH_DATOS) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) INPUT
DIRECT KEYED ENV(RECSIZE(LONGITUD));
CALL BUSCA(TESAURO,CLAVE,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,
NODO,PF_CB,VTA,ULT_N,POS_RE,INF);
CONT_AU = RANK(SUBSTR(ENTRADA,POS_ABS+5,1))*256 +
RANK(SUBSTR(ENTRADA,POS_ABS+6,1));
DIR = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+2,1)))*10000 +
RANK(SUBSTR(ENTRADA,POS_ABS+3,1))*100 +
RANK(SUBSTR(ENTRADA,POS_ABS+4,1));

END;   /*   BUSCA INDICE   */

GRABA_AUTOR: PROCEDURE;

NUMERO_ARC = 1;
I = 1;
DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_ARC);
   I = I + 1;
END;
LONGITUD = CONTROL_ARCH(I).LON_REG;
CLOSE FILE(ARCH_DATOS);
OPEN FILE(ARCH_DATOS) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) INPUT
DIRECT KEYED ENV(RECSIZE(LONGITUD));
K = 0;
J = 0;
CONTINUE:
J = J + 1;
BA_ENC = '0'B;
DO WHILE(BA_ENC='0'B);
   K = K + 1;
   CLAVE2 = CLAVE||CHAR(K);
 CALL BUSCA(TESAURO,CLAVE2,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
   PF_CB,VTA,ULT_N,POS_RE,INF);
END;
NUM_AUT = RANK(SUBSTR(ENTRADA,POS_ABS+5,1))*256 +
RANK(SUBSTR(ENTRADA,POS_ABS+6,1));
```

```
AP2 = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+7,1)))*10000 +
RANK(SUBSTR(ENTRADA,POS_ABS+8,1))*100 +
RANK(SUBSTR(ENTRADA,POS_ABS+9,1));
NUMERO_INDICE = 11;
I = 1;
DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
   I = I + 1;
END;
CLOSE FILE(INDICE);
OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) INPUT
DIRECT KEYED ENV(RECSIZE(1024));
CLAVE2 = CHAR(NUM_AUT);
CALL BUSCA(TESAURO,CLAVE2,KA_ENC,POS_ABS,NES_CM,PF_CM,LON_CM,NRO(
PF_CB,VIA,ULT_N,POS_RE,INF);
DIR = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+2,1)))*10000 +
RANK(SUBSTR(ENTRADA,POS_ABS+3,1))*100 +
RANK(SUBSTR(ENTRADA,POS_ABS+4,1));
READ FILE(ARCH_DATOS) INTO(BLQ_MONO_PP) KEY(DIR);
POS = 1;
I = 1;
REPI: LON = RANK(SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,POS,1));
      CLAVE2 = SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,POS+1,LON-1);
      POS = POS + LON;
      LON = RANK(SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,POS,1)) - 1;
      IF LON ^= 0 THEN
         CLAVE2 = CLAVE2!!SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,POS+1,LON);
      IF AP2 = 1 THEN DO;
         IF I = 1 THEN DO;
            I = I + 1;
            POS = POS + LON + 1;
            AUTOR = CLAVE2;
            GOTO REPI;
         END;
      END;
  IF DIRE_PANTA = 4 THEN DO;
     IF AP2 = 0 THEN DO;
        PUT FILE(PEDIDO_PROF) SKIP EDIT("PP:")(A);
        PUT FILE(PEDIDO_PROF) SKIP EDIT(CLAVE2)(A);
     END;
     ELSE DO;
        PUT FILE(PEDIDO_PROF) SKIP EDIT("LIBRO:")(A);
        PUT FILE(PEDIDO_PROF) SKIP EDIT(AUTOR)(A);
        PUT FILE(PEDIDO_PROF) SKIP EDIT(CLAVE2)(A);
     END;
     NUMERO_INDICE = 13;
END;
IF DIRE_PANTA = 6 THEN DO;
     IF AP2 = 0 THEN DO;
        PUT FILE(PEDIDO_IMP) SKIP EDIT("PP:")(A);
        PUT FILE(PEDIDO_IMP) SKIP EDIT(CLAVE2)(A);
     END;
     ELSE DO;
        PUT FILE(PEDIDO_IMP) SKIP EDIT("LIBRO:")(A);
        PUT FILE(PEDIDO_IMP) SKIP EDIT(AUTOR)(A);
        PUT FILE(PEDIDO_IMP) SKIP EDIT(CLAVE2)(A);
     END;
     NUMERO_INDICE = 15;
END;
IF DIRE_PANTA = 8 THEN DO;
     IF AP2 = 0 THEN DO;
        PUT FILE(PEDIDO_PAG) SKIP EDIT("PP:")(A);
        PUT FILE(PEDIDO_PAG) SKIP EDIT(CLAVE2)(A);
     END;
```

```
     ELSE DO;
        PUT FILE(PEDIDO_PAG) SKIP EDIT("LIBRO:")(A);
        PUT FILE(PEDIDO_PAG) SKIP EDIT(AUTOR)(A);
        PUT FILE(PEDIDO_PAG) SKIP EDIT(CLAVE2)(A);
     END;
     NUMERO_INDICE = 17;
END;
IF J < CONT_AU THEN DO;
    I = 1;
    DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
       I = I + 1;
    END;
    CLOSE FILE(INDICE);
    OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) INPUT
    DIRECT KEYED ENV(RECSIZE(1024));
    GOTO CONTINUE;
END;

END;    /*    GRABA_AUTOR     */
CLOSE FILE(PEDIDO_TMP);
CLOSE FILE(PEDIDO_PAG);
CLOSE FILE(PEDIDO_PROF);

END;   /*      MUESTRA_AUTOR    */
```

```
CONSULTA_OBSERVACION: PROCEDURE(J,CADENA_LEIDA);

/*******************************************************************
 *                                                                 *
 *     PROGRAMA QUE MUESTRA LAS OBSERVACIONES POR PANTALLA         *
 *                                                                 *
 *        PARAMETROS:                                              *
 *            J: NUMERO QUE INDICA QUE OBSERVACION SE QUIERE VER   *
 *               1: OBSERVACION SIGUIENTE                          *
 *               0: OBSERVACION ANTERIOR                           *
 *            CADENA_LEIDA: CADENA DE INFORMACION                  *
 *                                                                 *
 *        VARIABLES:                                               *
 *            OB: NUMERO DE OBSERVACIONES EN CONSULTA              *
 *                                                                 *
 *            LON: CONTIENE EL ULTIMO VALOR EN LA CADENA           *
 *                                                                 *
 *        LLAMADA POR:                                             *
 *            MAS_MENOS_LINEAS                                     *
 *                                                                 *
 *        LLAMA A:                                                 *
 *            GOTOXY                                               *
 *                                                                 *
 *        ARCHIVOS:                                                *
 *            OUT: ARCHIVO DE SALIDA                               *
 *                                                                 *
 *        MACRO ALGORITMO DEL PROGRAMA:                            *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *                                                                 *
 *******************************************************************/

DCL
CK                                              CHAR(1),
BA_ENC                                          BIT(1),
(OUT,IN)                                        FILE,
(J,POS,LON,K,I)                                 BIN FIXED,
CADENA_LEIDA                                    CHAR(1000) VAR,
(OB,CONT_AU_PROF,CONT_AU_FACT,CONT_AU_PED,
 CONT_AU_CHEQ,CONT_AU_PAG,CONT_AU_RECI)     BIN FIXED EXT STATIC,
(CONTADOR_PR,CONTADOR_PP,CONTADOR_AU)       BIN FIXED EXT STATIC,
GOTOXY ENTRY(BIN FIXED,BIN FIXED);

CK = ASCII(11);

IF J = 1 THEN DO;
```

```
    OR = OR + 1;
POS = LENGTH(CADENA_LEIDA);
LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    IF LON = 255 THEN DO;
       OR = OR - 1;
       CALL MENSAJE;
    END;
IF (CONT_AU_PROF ^= 0 | CONT_AU_FACT ^= 0 | CONT_AU_RECT ^= 0 | CONT_AU_PAG ^= 0
 |
CONT_AU_CHEQ ^= 0 | CONTADOR_PR ^= 0 | CONTADOR_PP ^= 0 | CONTADOR_AU ^= 0) THEN
 DO;
    POS = 1;
    LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    DO WHILE(LON ^= 255);
       POS = POS + LON;
       LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    END;
END;
IF CONT_AU_PED ^= 0 THEN DO;
    BA_ENC = "0"B;
    POS = 1;
    LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    DO WHILE(BA_ENC = "0"B);
       DO WHILE(LON ^= 253 & LON ^= 255);
          POS = POS + LON;
          LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
       END;
       IF LON = 255 THEN
          BA_ENC = "1"B;
       IF LON = 253 THEN DO;
          POS = POS + 1;
          LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
          POS = POS + LON + 1;
          LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
       END;
    END;
END;
K = 1;
CALL MUESTRA_OBS;
END;
IF J = 2 THEN DO;
    OR = OR - 1;
IF (CONT_AU_PROF ^= 0 | CONT_AU_FACT ^= 0 | CONT_AU_RECT ^= 0 | CONT_AU_PAG ^= 0
 |
CONT_AU_CHEQ ^= 0 | CONTADOR_PR ^= 0 | CONTADOR_PP ^= 0 | CONTADOR_AU ^= 0) THEN
 DO;
    POS = 1;
    LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    DO WHILE(LON ^= 255);
       POS = POS + LON;
       LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    END;
END;
IF CONT_AU_PED ^= 0 THEN DO;
    BA_ENC = "0"B;
    POS = 1;
    LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
    DO WHILE(BA_ENC = "0"B);
       DO WHILE(LON ^= 253 & LON ^= 255);
          POS = POS + LON;
          LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
       END;
       IF LON = 255 THEN
          BA_ENC = "1"B;
```

```
ION = RANK(SUBSTR(CADENA,POS,1));
```

```
            POS = POS + LON + 1;
            LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
         END;
      END;
   END;
K = 1;
CALL MUESTRA_OBS;
END;

MUESTRA_OBS: PROCEDURE;

SIGA:
   POS = POS + 1;
   LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
   IF LON = 250 THEN DO;
      OB = OB - 1;
      CALL MENSAJE;
   END;
   POS = POS + 1;
   LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
   IF K ^= OB THEN DO;
      K = K + 1;
      DO I = 1 TO 3;
         POS = POS + LON;
         LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
      END;
      GOTO SIGA;
   END;
   CALL GOTOXY(0,18);
   PUT FILE(OUT) EDIT("OBSERVACION:",CK)(A,A);
   DO I = 1 TO 3;
      CALL GOTOXY(15,17+I);
      PUT FILE(OUT) EDIT(SUBSTR(CADENA_LEIDA,POS+1,LON-1),CK)(A,A);
      POS = POS + LON;
      LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
   END;

 END;  /*      MUESTRA_OBS       */

MENSAJE: PROCEDURE;

     CALL GOTOXY(0,22);
     PUT FILE(OUT) EDIT("NO HAY MAS OBSERVACION",CK)(A,A);
     GOTO FIN1;
END;

FIN1:
END;     /*    CONSULTA_OBSERVACION    */
```

```
ADELANTA: PROCEDURE(K,J,GUAR_J,NUMERO_INDICE,DIRE_PANTA,F,
GUAR_K,CLAVE,TESAURO,NUM_AUT,AP2,DIR);

/*****************************************************************************
 *                                                                           *
 *  RUTINA QUE SE ENCARGA DE ADELANTAR EN LOS AUTORES/PP                     *
 *  DEL PROCESO ANTERIOR AL PROCESO QUE SE ESTA CREANDO                      *
 *                                                                           *
 *      PARAMETROS:                                                          *
 *                                                                           *
 *          K: INDICA EL NUMERO DE AUTOR DONDE SE ENCUENTRA EN EL            *
 *             PROCESO                                                       *
 *                                                                           *
 *          J: INDICA EL NUMERO DE AUTORES QUE HAN SIDO PROCESADOS           *
 *                                                                           *
 *          GUAR_J: NUMERO QUE INDICA EN QUE VALOR DE J ESTARA               *
 *          ANTES DE RETROCEDER                                              *
 *                                                                           *
 *          GUAR_K: NUMERO QUE INDICA EN QUE VALOR DE K ESTARA               *
 *          ANTES DE RETROCEDER                                              *
 *                                                                           *
 *          NUMERO_INDICE: NUMERO DEL INDICE CORRESPONDIENTE AL              *
 *          PROCESO DONDE SE ESTA RETROCEDIENDO                              *
 *                                                                           *
 *          CLAVE: NOMBRE DEL PROCESO QUE SE ESTA REALIZANDO                 *
 *                                                                           *
 *          NUM_AUT: NUMERO DEL AUTOR/PP CORRESPONDIENTE                     *
 *                                                                           *
 *          AP2: INDICA SI SE TRATA DE UN AUTOR O UNA PP                     *
 *               1: AUTOR                                                    *
 *               0: PP                                                       *
 *                                                                           *
 *          DIR: DIRECCION DEL REGISTRO EN EL ARCHIVO DE                     *
 *          AUTOR_PP                                                         *
 *                                                                           *
 *      VARIABLES:                                                           *
 *                                                                           *
 *          NUMERO_INDICE: NUMERO DEL INDICE DEL PROCESO ANTERIOR            *
 *                                                                           *
 *      LLAMADA POR:                                                         *
 *          MUESTRA_INFORMACION                                              *
 *                                                                           *
 *      LLAMA A:                                                             *
 *          BUSCA                                                            *
 *                                                                           *
 *      ARCHIVOS:                                                            *
 *          CONTROL_ARCHIVOS: CONTIENE INFORMACION DE LOS PRIN-              *
 *          CIPALES ARCHIVOS DEL SISTEMA                                     *
 *                                                                           *
 *****************************************************************************/

DCL
(I,NUM_AUT,J,K,GUAR_K,GUAR_J,NUMERO_INDICE,DIRE_PANTA) BIN FIXED,
(AP2,DIR)                                              DEC FIXED,
(CLAVE2,CLAVE)                                         CHAR(120) VAR,
(INDICE,CONTROL_ARCHIVO)                               FILE,
TESAURO                                                FILE VARIABLE,
(BA_ENC,F)                                             BIT(1),
(POS_ARS,POS,PP_CR,ION_CR,PP_CR,UNI_N,POS_RF,VTA(5))   BIN FIXED,
RES_CR                                                 CHAR(120) VAR,
01 INF,
   02 APU_RAIZ                                         BIN FIXED,
   02 NO_NIV                                           BIN FIXED,
```

```
   02 NO_NODO                                                    BIN FIXED,
   02 APU_PROX_VACIO                                            BIN FIXED,
01 NODO,
   02 CAR_NODO,
      03 NUM_N                                                  BIN FIXED,
      03 NTU                                                    BIN FIXED,
      03 NC                                                     BIN FIXED,
      03 TN                                                     BIN FIXED,
      03 VECINA_I                                               BIN FIXED,
      03 VECINA_D                                               BIN FIXED,
   02 ENTRADA                                                   CHAR(1009) VAR,
01 CONTROL_ARCH(16),
   02 TIPO                                                      BIN FIXED,
   02 N_ARC                                                     BIN FIXED,
   02 LON_REG                                                   BIN FIXED,
   02 ULT_VACIO                                                 BIN FIXED,
   02 MAX_DIR                                                   BIN FIXED,
   02 NOMBRE_ARC                                                CHAR(20) VAR,
BUSCA_ENTRY(FILE VARIABLE,CHAR(120) VAR,BIT(1),BIN FIXED,CHAR(120) VAR,
BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,BIN FIXED,
(5) BIN FIXED,BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,
2 BIN FIXED,2 BIN FIXED);

READ FILE(CONTROL_ARCHIVO) INTO(CONTROL_ARCH) KEY(1);

      K = K + 1;
      J = J + 1;
      IF J <= GUAR_J THEN DO;
         IF (NUMERO_INDICE=12 | NUMERO_INDICE=13 | NUMERO_INDICE=14 | NUMERO_IND
CE = 15 | NUMERO_INDICE = 17)THEN
            GOTO RA;
         IF DIRE_PANTA = 4 THEN
            NUMERO_INDICE = 12;
         IF DIRE_PANTA = 5 THEN
            NUMERO_INDICE = 13;
         IF DIRE_PANTA = 6 THEN
            NUMERO_INDICE = 14;
         IF DIRE_PANTA = 8 THEN DO;
            IF F = "0"B THEN
               NUMERO_INDICE = 14;
            ELSE
               NUMERO_INDICE = 15;
         END;
         IF DIRE_PANTA = 9 THEN
            NUMERO_INDICE = 17;
         CALL BUSCA_INDICE;
      RA:BA_ENC = "0"B;
         DO WHILE(BA_ENC = "0"B & K <= GUAR_K);
            CLAVE2 = CLAVE1||CHAR(K);
            CALL BUSCA(TESAURO,CLAVE2,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
            PF_CR,VTA,ULT_N,POS_RE,INF);
            IF BA_ENC = "0"B THEN
               K = K + 1;
         END;
         NUM_AUX = RANK(SUBSTR(ENTRADA,POS_ABS+5,1))*256 +
         RANK(SUBSTR(ENTRADA,POS_ABS+6,1));
         AP2 = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+7,1)))*10000 +
         RANK(SUBSTR(ENTRADA,POS_ABS+8,1))*100 +
         RANK(SUBSTR(ENTRADA,POS_ABS+9,1));
         NUMERO_INDICE = 11;
         CALL BUSCA_INDICE;
         CLAVE2 = CHAR(NUM_AUX);
         CALL BUSCA(TESAURO,CLAVE2,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
```

```
          PF CR,VIA,ULT_N,POS_RE,TNF);
          DTR = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+2,1)))*10000 +
          RANK(SUBSTR(ENTRADA,POS_ABS+3,1))*100 +
          RANK(SUBSTR(ENTRADA,POS_ABS+4,1));
       END;

BUSCA_INDICE: PROCEDURE;

    CLOSE FILE(INDICE);
    I = 1;
    DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
        I = I + 1;
    END;
    OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
    DIRECT KEYED ENV(RECSIZE(1024));

END; /*  BUSCA INDICE  */

END;    /*       ADELANTA      */
```

```
RETROCEDE: PROCEDURE(PRI,J,K,GUAR_K,GUAR_J,NUMERO_INDICE,
DIRE_PANTA,F,CLAVE,TESAURO,NUM_AUT,AP?,DIR,RR);

/*****************************************************************************
 *                                                                           *
 *    RUTINA QUE SE ENCARGA DE RETROCEDER EN LOS AUTORES/PP                  *
 *    DEL PROCESO ANTERIOR AL PROCESO QUE SE ESTA CREANDO                    *
 *                                                                           *
 *        PARAMETROS:                                                        *
 *                                                                           *
 *           PRI: INDICA SI SE ESTA INICIANDO EL PROCESO                     *
 *                DEL RETROCESO                                              *
 *                0: ES ESTA INICIANDO EL PROCESO(SE GUARDA EL               *
 *                   NUMERO DE AUTOR A PARTIR DEL CUAL CONTEN/A              *
 *                   EL RETROCESO                                            *
 *                1: EL RETROCESO YA HABIA SIDO INICIADO                     *
 *                                                                           *
 *                                                                           *
 *           J: INDICA EL NUMERO DE AUTORES Y/O PP QUE HAN SIDO              *
 *              PROCESADOS                                                   *
 *                                                                           *
 *           K: INDICA EL NUMERO DE AUTOR DONDE SE ENCONTRABA                *
 *              ANTES DE RETROCEDER                                          *
 *                                                                           *
 *           GUAR_J: GUARDA EL VALOR DE J                                    *
 *                                                                           *
 *           GUAR_K: GUARDA EL VALOR DE K                                    *
 *                                                                           *
 *           NUMERO_INDICE: NUMERO DEL INDICE DEL PROCESO DONDE              *
 *           SE ESTA RETROCEDIENDO                                           *
 *                                                                           *
 *           CLAVE: NOMBRE DEL PROCESO                                       *
 *                                                                           *
 *           NUM_AUT: NUMERO DEL AUTOR/PP CORRESPONDIENTE                    *
 *                                                                           *
 *           AP?: INDICADOR DE AUTOR O PP                                    *
 *                1: AUTOR                                                   *
 *                0: PP                                                      *
 *                                                                           *
 *           DIR: DIRECCION DEL REGISTRO CORRESPONDIENTE AL                  *
 *                AUTOR/PP                                                   *
 *                                                                           *
 *        VARIABLES:                                                         *
 *                                                                           *
 *           NUMERO_INDICE: NUMERO DEL INDICE CORRESPONDIENTE                *
 *           AL PROCESO EN RETROCESO                                         *
 *                                                                           *
 *        LLAMADA POR:                                                       *
 *                                                                           *
 *           MUESTRA_INFORMACION                                             *
 *                                                                           *
 *        LLAMA A:                                                           *
 *           BUSCA                                                           *
 *                                                                           *
 *        ARCHIVOS:                                                          *
 *                                                                           *
 *           CONTROL_ARCHIVOS: CONTIENE INFORMACION DE LOS PRIN-             *
 *           CIPALES ARCHIVOS DEL SISTEMAS                                   *
 *                                                                           *
 *****************************************************************************/

DCL
RR                                                    BIN FIXED,
```

```
NUM_AUT                                                    BIN FIXED,
CLAVE                                                      CHAR(120) VAR,
(PRT,BA_ENC,F)                                              BIT(1),
(J,K,GUAR_J,GUAR_K)                                         BIN FIXED,
(NUMERO_INDICE,DIRE_PANTA,I)                                BIN FIXED,
(POS_ABS,PF_CM,LON_CM,PF_CB,ULT_N,POS_RE,VIA(5))            BIN FIXED,
RES_CM                                                     CHAR(120) VAR,
(INDICE,ARCH_DATOS,CONTROL_ARCHIVO)                         FILE,
TESAURO                                                    FILE VARIABLE,
01 INF,
   02 APU_RAIZ                                             BIN FIXED,
   02 NO_NIV                                               BIN FIXED,
   02 NO_NODO                                              BIN FIXED,
   02 APU_PRIX_VACIO                                       BIN FIXED,
01 NODO,
   02 CAR_NODO,
      03 NUM_N                                             BIN FIXED,
      03 NIV                                               BIN FIXED,
      03 NC                                                BIN FIXED,
      03 IN                                                BIN FIXED,
      03 VECINA_I                                          BIN FIXED,
      03 VECINA_D                                          BIN FIXED,
   02 ENTRADA                                             CHAR(1009) VAR,
01 BLQ_MONO_PP,
   02 CADENA_LEIDA2                                        CHAR(1000) VAR,
01 CONTROL_ARCH(14),
   02 TIPO                                                 BIN FIXED,
   02 N_ARC                                                BIN FIXED,
   02 LON_REG                                              BIN FIXED,
   02 ULT_VACIO                                            BIN FIXED,
   02 MAX_DIR                                              BIN FIXED,
   02 NOMBRE_ARC                                           CHAR(20) VAR,
01 CONTROL_BLQ_MONO(25),
   02 CADENA_MONO                                          CHAR(10) VAR,
   02 POSICION_MONO                                        BIN FIXED,
01 CONTROL_BLQ_PP(25),
   02 CADENA_PP                                            CHAR(10) VAR,
   02 POSICION_PP                                          BIN FIXED,
(CLAVE2,AUTOR)                                             CHAR(120) VAR,
(AP2,DIR)                                                  DEC FIXED,
BUSCA_ENTRY(FILE VARIABLE,CHAR(120) VAR,BIT(1),BIN FIXED,CHAR(120) VAR,
BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,BIN FIXED,
(5) BIN FIXED,BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,
2 BIN FIXED,2 BIN FIXED);

READ FILE(CONTROL_ARCHIVO) INTO(CONTROL_ARCH) KEY(I);

   IF PRT = "0"B THEN DO;
      PRT = "1"B;
      GUAR_K = K;
      GUAR_J = J;
   END;
   J = J - 1;
   IF J = 0 THEN DO;
      J = 1;
      GOTO FIN1;
   END;
   IF DIRE_PANTA = 8 THEN DO;
      IF J = RR THEN DO;
         J = J + 1;
         GOTO FIN1;
      END;
```

```
   END;
   IF (NUMERO_INDICE=12 | NUMERO_INDICE=13 | NUMERO_INDICE=14 | NUMERO_INDICE =
15 | NUMERO_INDICE = 17)THEN
      GOTO AP;
   IF DIRF_PANTA = 4 THEN
      NUMERO_INDICE = 12;
   IF DIRF_PANTA = 5 THEN
      NUMERO_INDICE = 13;
   IF DIRF_PANTA = 6 THEN
      NUMERO_INDICE = 14;
   IF DIRF_PANTA = 8 THEN DO;
      IF F = "0"B THEN
         NUMERO_INDICE = 14;
      ELSE
         NUMERO_INDICE = 15;
   END;
   IF DIRF_PANTA = 9 THEN
      NUMERO_INDICE = 17;
   CALL BUSCA_INDICE;
AP:K = K - 1;
   BA_ENC = "0"B;
   DO WHILE(BA_ENC = "0"B & K ^= 0);
      CLAVE2 = CLAVE||CHAR(K);
      CALL BUSCA(TESAURO,CLAVE2,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
      PF_CB,VTA,DET_N,POS_RE,INF);
      IF BA_ENC = "0"B THEN
         K = K - 1;
   END;
   NUM_AUX = RANK(SUBSTR(ENTRADA,POS_ABS+5,1))*256 +
   RANK(SUBSTR(ENTRADA,POS_ABS+6,1));
   AP2 = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+7,1)))*10000 +
   RANK(SUBSTR(ENTRADA,POS_ABS+8,1))*100 +
   RANK(SUBSTR(ENTRADA,POS_ABS+9,1));
   CLAVE2 = CHAR(NUM_AUX);
   NUMERO_INDICE = 11;
   CALL BUSCA_INDICE;
   CALL BUSCA(TESAURO,CLAVE2,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
   PF_CB,VTA,DET_N,POS_RE,INF);
   DIR = DEC(RANK(SUBSTR(ENTRADA,POS_ABS+2,1)))*10000 +
   RANK(SUBSTR(ENTRADA,POS_ABS+3,1))*100 +
   RANK(SUBSTR(ENTRADA,POS_ABS+4,1));

/*****************************************************************************/

BUSCA_INDICE: PROCEDURE;

   CLOSE FILE(INDICE);
   I = 1;
   DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
      I = I + 1;
   END;
   OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
   DIRECT KEYED ENV(RECSIZE(1024));

END; /*  BUSCA_INDICE  */

FIN1:

END;    /*     RETROCEDE    */
```

```
ELIMINA_ARB: PROCEDURE(TESAURO,RES_CB);

/*****************************************************************************
*                                                                            *
*                          ELIMINA                                           *
*                                                                            *
*     RUTINA QUE ELIMINA UNA CLAVE EN EL ARBOL                               *
*                                                                            *
*     RUTINAS QUE LLAMA:                                                     *
*                          BUSCA                                             *
*                                                                            *
*     LLAMADA POR       :                                                    *
*                                                                            *
*     PARAMETROS DE ENTRADA:                                                 *
*                                  TESAURO, RES_CB                           *
*****************************************************************************

       DECLARACION DE VARIABLES */
DCL
    (POS_ABS,POS_RE,PF_CB,PF_CM,LON_REG_PAR,LON_CM)  BIN FIXED,
    (BA_ENC,BA_DEC,IGUAL)                            BIT(1),
PANTALLA                                 FILE,
    (RES_CB,RES_CM,CLAVE,RES_CE)                     CHAR(120) VAR,
    (LON_CB,APH,VIA(5),ULT_N,PF_CE)                  BIN FIXED,
    (AP,APL)                                         DEC FIXED,
    (PF_CM_S,LON_CM_S,LON_CLA)                       BIN FIXED,
    1 INF,
      2 APU_KATZ                                     BIN FIXED,
      2 NO_NIV                                       BIN FIXED,
      2 NO_NODO                                      BIN FIXED,
      2 APU_PROX_VACIO                               BIN FIXED,
    1 NODO,
      2 CAB_NODO,
        3 NUM_N                                      BIN FIXED,
        3 NIV                                        BIN FIXED,
        3 NC                                         BIN FIXED,
        3 TN                                         BIN FIXED,
        3 VECINA_I                                   BIN FIXED,
        3 VECINA_D                                   BIN FIXED,
      2 ENTRADA                                      CHAR(1009) VAR,
    TESAURO                                          FILE VARIABLE;
DCL
   BUSCA  ENTRY(FILE VARIABLE,CHAR(120)VAR,BIT(1),BIN FIXED,
          CHAR(120)VAR, BIN FIXED, BIN FIXED),1,2,3 BIN FIXED,
          3 BIN FIXED,3 BIN FIXED,3 BIN FIXED, 3 BIN FIXED,
          3 BIN FIXED),2 CHAR(1009) VAR, BIN FIXED),(5) BIN FIXED,
          BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,2 BIN FIXED,
          2 BIN FIXED);
/*              PROGRAMA PRINCIPAL                    */
CALL BUSCA(TESAURO,RES_CB,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,NODO,
          PF_CB,VIA,ULT_N,POS_RE,INF);
PUT FILE(PANTALLA) SKIP LIST(BA_ENC,RES_CM);
LON_REG_PAR = 10;
IF BA_ENC ='1'B THEN DO;
PUT FILE(PANTALLA) SKIP LIST(BA_ENC,RES_CM);
 IF POS_RE = NC THEN DO;
  IF NC = 1 THEN DO;
  CALL DEVUELVE_VACIO(VIA(ULT_N),VECINA_I,VECINA_D,NUM_N);
  DO WHILE(NIV+1(NO_NIV);
   ULT_N=ULT_N-1;
   READ FILE(TESAURO) INTO(NODO) KEY(VIA(ULT_N));
   IF NC=1 THEN
    CALL DEVUELVE_VACIO(VIA(ULT_N),VECINA_I,VECINA_D,NUM_N);
```

```
    ELSE DO;
     CALL BUS_DIREC(VIA,ULT_N,POS_ABS,PF_CE,LON_CM,POS_RE,LON_REG_PAR);
     IF POS_RE=NC THEN DO;
      ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1);
      TN = TN - LON_CM;
     END;
     ELSE DO;
      RES_CE = SUBSTR(ENTRADA,POS_ABS+LON_REG_PAR,LON_CM-LON_REG_PAR);
      CALL ELIN_CLAVE(PF_CE,RES_CE,POS_RE,LON_CM,LON_REG_PAR,NODO,
      POS_ABS);
     END;
     NC = NC - 1;
     GOTO TERMINA;
   END;
END;
IF NTV ^= 1 THEN DO;
 ULT_N = ULT_N-1;
 READ FILE(TESAURO) INTO(NODO) KEY(VIA(ULT_N));
 CALL BUS_DIREC(VIA,ULT_N,POS_ABS,PF_CE,LON_CM,POS_RE,LON_REG_PAR);
 IF NC-1 = 1 THEN DO;
  CALL DEVUELVE_VACIO(VIA(ULT_N),VECINA_I,VECINA_D,NUM_N);
  IF POS_RE = NC THEN
   APU_RAIZ = RANK(SUBSTR(ENTRADA,3,1))*25A +
              RANK(SUBSTR(ENTRADA,4,1));
  ELSE
   APU_RAIZ = RANK(SUBSTR(ENTRADA,POS_ABS+LON_CM+2,1))*25A +
              RANK(SUBSTR(ENTRADA,POS_ABS+LON_CM+3,1));
  NO_NTV = NO_NTV-1;
  REWRITE FILE(TESAURO) FROM(INF) KEY(1);
  RETURN;
END;
ELSE DO;
 IF POS_RE = NC THEN DO;
  ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1);
  TN = TN-LON_CM;
 END;
 ELSE DO;
  RES_CE = SUBSTR(ENTRADA,POS_ABS+LON_REG_PAR,LON_CM-LON_REG_PAR);
  CALL ELIN_CLAVE(PF_CE,RES_CE,POS_RE,LON_CM,LON_REG_PAR,NODO,
  POS_ABS);
 END;
 NC = NC-1;
END;
END;
END;
ELSE DO;
 ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1);
 NC = NC-1;
 TN = TN-LON_CM;
END;
END;
ELSE DO;
 CALL ELIN_CLAVE(PF_CM,RES_CM,POS_RE,LON_CM,LON_REG_PAR,NODO,POS_ABS);
NC = NC - 1;
END;
TERMINA:
REWRITE FILE(TESAURO) FROM(NODO) KEY(VIA(ULT_N));
END;
ELSE DO;
 PUT FILE(PANTALLA) SKIP EDIT("NO SE ENCONTRO LA CLAVE")(A);
 PUT FILE(PANTALLA) SKIP LIST(BA_ENC,RES_CM);
END;
/*************************************************************************
```

```
 *                                                                    *
 *                          BUS DIREC                                 *
 *     RUTINA QUE BUSCA LA DIRECCION DE LA CLAVE A ELIMINAR           *
 *                                                                    *
 **********************************************************************/
BUS_DIREC: PROCEDURE(VTA,ULT_N,POS_ABS,PF_CF,
                     LON_CM,POS_RE, LON_REG_PAR);
DCL  (VTA(5),ULT_N,POS_ABS,PF_CF,LON_CM,POS_RE,APN)       BIN FIXED,
       LON_REG_PAR                                        BIN FIXED;
POS_ABS=1;
POS_RE=1;
LON_CM = RANK(SUBSTR(ENTRADA,POS_ABS+1,1));
LON_REG_PAR=4;
APN=RANK(SUBSTR(ENTRADA,POS_ABS+2,1))*256+
    RANK(SUBSTR(ENTRADA,POS_ABS+3,1));
DO WHILE(VTA(ULT_N+1) ^= APN);
 RES_CB = SUBSTR(ENTRADA,POS_ABS+LON_REG_PAR,LON_CM-LON_REG_PAR);
 POS_ABS = POS_ABS+LON_CM;
 APN = RANK(SUBSTR(ENTRADA,POS_ABS+2,1))*256+
     RANK(SUBSTR(ENTRADA,POS_ABS+3,1));
 LON_CM=RANK(SUBSTR(ENTRADA,POS_ABS+1,1));
 POS_RE = POS_RE+1;
END;
PF_CF=RANK(SUBSTR(ENTRADA,POS_ABS,1));
END;/* FIN DE BUSCA DIRECCION*/
/**********************************************************************
 *                     ELIM_CLAVE                                     *
 *    RUTINA QUE SE ENCARGA DE ELIMINAR LA CLAVE                      *
 *                                                                    *
 **********************************************************************/
ELIM_CLAVE: PROCEDURE(PF_CM,RES_CM,POS_RE,LON_CM,LON_REG_PAR,
                      NODO,POS_ABS);
DCL  (PF_CM,LON_REG_PAR,POS_RE,LON_CM,POS_ABS)        BIN FIXED,
      API                                             CHAR(8)VAR,
     RES_CM                  CHAR(120)VAR,
     1 NODO,
       2 CAB_NODO,
         3 NUM_N             BIN FIXED,
         3 NTV               BIN FIXED,
         3 NC                BIN FIXED,
         3 TN                BIN FIXED,
         3 VECINA_I          BIN FIXED,
         3 VECINA_D          BIN FIXED,
       2 ENTRADA             CHAR(1009)VAR;
PF_CM_S = RANK(SUBSTR(ENTRADA,POS_ABS+LON_CM,1));
IF PF_CM_S-PF_CM > 0 THEN DO;
 LON_CM_S=RANK(SUBSTR(ENTRADA,POS_ABS+LON_CM+1,1));
 PF_CM_S = PF_CM_S-PF_CM;
 CLAVE=SUBSTR(RES_CM,1,PF_CM_S)||SUBSTR(ENTRADA,
 POS_ABS+LON_CM+LON_REG_PAR,LON_CM_S-LON_REG_PAR);
PUT FILE(PANTALLA) SKIP LIST(POS_ABS,LON_CM,LON_REG_PAR);
 LON_CLA = LON_CM_S + PF_CM_S;
 API = SUBSTR(ENTRADA,POS_ABS+LON_CM+2,LON_REG_PAR-2);
 IF POS_ABS > 1 THEN DO;
  IF POS_RE = NC-1 THEN
  ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1)||ASCII(PF_CM)||ASCII(LON_CLA)
            ||API||CLAVE;
  ELSE
   ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1)||ASCII(PF_CM)||ASCII(LON_CLA)
             ||API||CLAVE||SUBSTR(ENTRADA,POS_ABS+LON_CM+LON_CM_S);
  TN = TN-LON_CM+PF_CM_S;
 END;
 ELSE DO;
```

```
   IF NC = 2  THEN
    ENTRADA = ASCII(PF_CM)||ASCII(LON_CLA)||APL||CLAVE;
   ELSE
    ENTRADA = ASCII(PF_CM)||ASCII(LON_CLA)||APL||CLAVE||SUBSTR(ENTRADA,
              POS_ABS+LON_CM+LON_CM_S);
   TN = TN-LON_CM+PF_CM_S;
  END;
 END;
 ELSE DO;
  IF POS_ABS > 1 THEN DO;
   ENTRADA = SUBSTR(ENTRADA,1,POS_ABS-1)||SUBSTR(ENTRADA,POS_ABS+LON_CM);
   TN = TN-LON_CM;
  END;
  ELSE DO;
   ENTRADA = SUBSTR(ENTRADA,POS_ABS+LON_CM);
   TN = TN-LON_CM;
  END;
 END;
END;/* FIN DE ELIMINA CLAVE*/
/*****************************************************************
*                                                                *
*                        DEVUELVE_VACIO                          *
*                                                                *
*     RUTINA QUE DEVUELVE UN VACIO Y ACTUALIZA LOS               *
*                    VECINOS DE UN NODO                          *
*                                                                *
*****************************************************************/
DEVUELVE_VACIO: PROCEDURE(DTR,VECINA_I,VECINA_D,NUM_N);
DCL       (DTR,NUM_N)                                  BIN FIXED,
     (VECINO_I,VECINO_D,VECINA_I,VECINA_D)             BIN FIXED;
VECINO_I = VECINA_I;
VECINO_D = VECINA_D;
 READ FILE(TESAURO) INTO(INF) KEY(1);
 IF APU_PROX_VACIO = 0 THEN DO;
  APU_PROX_VACIO = DTR;
  NUM_N = 0;
 END;
 ELSE DO;
  NUM_N = APU_PROX_VACIO;
  APU_PROX_VACIO = DTR;
 END;
 NO_NODO = NO_NODO-1;
REWRITE FILE(TESAURO) FROM(INF) KEY(1);
REWRITE FILE(TESAURO) FROM(NODO) KEY(DTR);
IF VECINO_I ^= 0 THEN DO;
 READ FILE(TESAURO) INTO(NODO) KEY(VECINO_I);
 VECINA_D = VECINO_D;
 REWRITE FILE(TESAURO) FROM(NODO) KEY(VECINO_I);
END;
IF VECINO_D ^= 0 THEN DO;
 READ FILE(TESAURO) INTO(NODO) KEY(VECINO_D);
 VECINA_I = VECINO_I;
 REWRITE FILE(TESAURO) FROM(NODO) KEY(VECINO_D);
END;
END;/*     FIN DE DEVUELVE_VACIO        */
END;/* FIN DE ELIMINA*/
```

```
/*****************************************************************************
*                                                                           *
*  GENERA:                                                                  *
*             RutINa encargada de generar una pantalla basado en la         *
*  INformacion contenida en el archivo PANTA.                               *
*                                                                           *
*  Parametros: PANTA: Archivo variable que contiene las esperifica-         *
*                     ciones de la pantalla a generar.                      *
*                                                                           *
*  Llamada por: PRINCIPAL, INCORPURAR, MODIFICAR, LISTAR, INFORMAR          *
*                                                                           *
*  Llama a: GOTOXY                                                          *
*                                                                           *
*  Variables: ESTRUCTURA Estructura de datos que contINe la INforma-        *
*             cion de la pantalla a generar.                                *
*             ESTRUCTURA.REP cuantas lINeas hay que mostrar.                *
*             ESTRUCTURA.TABLA Contiene la INformacion de columna fi-       *
*             la y texto a escribir por pantalla.                           *
*                                                                           *
*  Archivos:  CONSOLA, PANTA(puede ser TABLA_PRIN, TABLA_INC, TABLA_-       *
*                            INC_H):                                        *
*                                                                           *
*****************************************************************************/

GENERA: PROCEDURE(NUM_PANTA,IV,FV);
DCL      PANTALLAS                          FILE;
DCL     (IV,FV,NUM_PANTA,POS)               BIN FIXED;
DCL     01 ESTRUCTURA,
             02 TIRA                        CHAR(1000) VAR;
DCL          (J,K)                          BIN FIXED,
             CK                             CHAR(1);
DCL          GOTOXY                         ENTRY(BIN FIXED,BIN FIXED);
DCL          (OUT,IN)                       FILE;
DCL
             (X,Y,LON)                      BIN FIXED,
              TEX1                          CHAR(80) VAR;
READ FILE(PANTALLAS) INTO (ESTRUCTURA) KEY(NUM_PANTA);
CK=ASCII(11);
POS = 1;
DO WHILE(SUBSTR(TIRA,POS,1) ^= ASCII(255));
   X = RANK(SUBSTR(TIRA,POS,1));
   Y = RANK(SUBSTR(TIRA,POS+1,1));
   LON = RANK(SUBSTR(TIRA,POS+2,1));
   TEX1 = SUBSTR(TIRA,POS+3,LON);
   CALL GOTOXY(X,Y);
   PUT FILE(OUT) EDIT(TEX1)(A);
   POS = POS + LON + 3;
END;
END;   /*    GENERA     */
```

```
/*****************************************************************************

    CORRIGE:
            Elimina todos los blancos existentes de mas en una cadena
    y busca la posicion en que se puede partir la cadena si esta es may-
    or de 10 caracteres.

    Parametros: AUX:    cadena a corregir.
                K:      posicion en que se puede dividir la cadena.

    Llamada por:  NUEVO_PR,NUEVO_AU,PEDIDO,MOD_TIT_AU.

    Llama a:

    Variables:  I: posicion de dos blancos consecutivos.

    Archivos:
*****************************************************************************/


CORRIGE: PROCEDURE(AUX,K);

DCL     AUX                         CHAR(120) VAR,
        (I,K)                       BIN FIXED;

I = INDEX(AUX,"  ");
DO WHILE(I^=0);
   IF I=1 THEN AUX=SUBSTR(AUX,3);
      ELSE AUX=SUBSTR(AUX,1,(I-1))!!SUBSTR(AUX,I+1);
   I = INDEX(AUX,"  ");
END;
K=71;
IF LENGTH(AUX) > 71 THEN
   DO WHILE(SUBSTR(AUX,K,1)^=" ");
      K=K-1;
   END;

END;    /*  CORRIGE */
```

```
BUSCA_BANCO: PROCEDURE(BANC,ENCONTRADO);

/*****************************************************************************
 *                                                                           *
 *  PROGRAMA ENCARGADO DE BUSCA UN BANCO EN EL  ARCHIVO                      *
 *  DE BANCOS                                                                *
 *                                                                           *
 *  PARAMETROS:                                                              *
 *                                                                           *
 *      BANC: NUMERO DEL BANCO A BUSCAR                                      *
 *      ENCONTRADO: INDICA SI EL  BANCO FUE ENCONTRADO                       *
 *                  1:BANCO EXISTE                                           *
 *                  0: NO EXISTE                                             *
 *                                                                           *
 * VARIABLES:                                                                *
 *                                                                           *
 * LLAMA A:                                                                  *
 *                                                                           *
 * LLAMADO POR:                                                              *
 *                                                                           *
 *      MUESTRA_DATOS                                                        *
 *      CREA_CHEQUE                                                          *
 *                                                                           *
 * ARCHIVOS:                                                                 *
 *                                                                           *
 *      BANCOS:CONTIENE TODOS LOS BANCOS DEL  SISTEMA                        *
 *****************************************************************************/

DCL
ENCONTRADO                              BIT(1),
BANC                                    BIN FIXED,
(BANCOS,IN,OUT)                         FILE,
I                                       BIN FIXED,
01 INFBAN,
   02 BLA                                                   BIN FIXED,
   02 BLS                                                   BIN FIXED,
   02 CONT_B                                                BIN FIXED,
   02 NUMERO_B(3)                                           BIN FIXED,
   02 SITUACION(3)                                          CHAR(1),
   02 NOMBRE_B(3)                                           CHAR(30) VAR,
   02 DIR_B(3)                                              CHAR(120) VAR;

OPEN FILE(BANCOS) RECORD UPDATE DIRECT KEYED;

READ FILE(BANCOS) INTO(INFBAN) KEY(1);
ENCONTRADO = "0"B;
I = 0;
DO WHILE(I < CONT_B & ENCONTRADO = "0"B);
      IF (BANC = NUMERO_B(I+1) & SITUACION(I+1) = "")THEN
         ENCONTRADO = "1"B;
      I = I + 1;
END;
DO WHILE(BLS ^= 0 & ENCONTRADO = "0"B);
   READ FILE(BANCOS) INTO(INFBAN) KEY(BLS);
   I = 0;
   DO WHILE(I < CONT_B & ENCONTRADO = "0"B);
      IF (BANC = NUMERO_B(I+1) & SITUACION(I+1) = "")THEN
         ENCONTRADO = "1"B;
      I = I + 1;
   END;
END;

END;    /*     BUSCA_BANCOS        */
```

```
BUSCA_MONEDA: PROCEDURE(MONED,ENCONTRADO);

/*****************************************************************************
 *                                                                           *
 *  PROGRAMA ENCARGADO DE BUSCAR UNA MONEDA EN EL  ARCHIVO                   *
 *  DE MONEDAS                                                               *
 *                                                                           *
 *  PARAMETROS:                                                              *
 *                                                                           *
 *     MONED: CONTIENE EL SIMBOLO DE LA MONEDA A BUSCAR                      *
 *                                                                           *
 *     ENCONTRADO: INDICA SI LA MONEDA A SIDO ENCONTRADA                     *
 *                                                                           *
 *  VARIABLES:                                                               *
 *                                                                           *
 *  LLAMADA POR:                                                             *
 *                                                                           *
 *     MUESTRA_DATOS                                                         *
 *                                                                           *
 *     CREA_CHEQUE                                                           *
 *                                                                           *
 *  LLAMA A:                                                                 *
 *                                                                           *
 *  ARCHIVOS:                                                                *
 *                                                                           *
 *     MONEDAS: CONTIENE LAS MONEDAS DEL  SISTEMA                            *
 *                                                                           *
 *****************************************************************************/

DCL
J                                                       BIN FIXED,
(MONEDAS,IN,OUT)                                        FILE,
MONED                                                   CHAR(5) VAR,
ENCONTRADO                                              BIT(1),
01  INFMON,
    02 BI_A                                             BIN FIXED,
    02 BI_S                                             BIN FIXED,
    02 CONT_M                                           BIN FIXED,
    02 SITUACION_M(12)                                  CHAR(1) VAR,
    02 MONE(12)                                         CHAR(10) VAR,
    02 SIMBOLO(12)                                      CHAR(5) VAR;

OPEN FILE(MONEDAS) RECORD UPDATE DIRECT KEYED
ENV(RECSIZE(1009));

READ FILE(MONEDAS) INTO(INFMON) KEY(1);
ENCONTRADO = "0"B;
J = 0;
DO WHILE(J < CONT_M & ENCONTRADO = "0"B);
       IF (MONED = SIMBOLO(J+1) & SITUACION_M(J+1) = "")THEN
          ENCONTRADO = "1"B;
       J = J + 1;
END;
DO WHILE(BI_S ^= 0 & ENCONTRADO = "0"B);
   READ FILE(MONEDAS) INTO(INFMON) KEY(BI_S);
   J = 0;
   DO WHILE(J < CONT_M & ENCONTRADO = "0"B);
       IF (MONED = SIMBOLO(J+1) & SITUACION_M(J+1) = "")THEN
          ENCONTRADO = "1"B;
       J = J + 1;
   END;
END;
```

```
END;    /*      MUSICA MONEDAS      */
```

```
CREA_OBSERVACION: PROCEDURE;

/*****************************************************************************
 *                                                                           *
 *  RUTINA QUE CREA LAS OBSERVACIONES                                        *
 *                                                                           *
 *      PARAMETROS:                                                          *
 *                                                                           *
 *      VARIABLES:                                                           *
 *           VALOR: TEXTO DE LA OBSERVACION                                  *
 *           VECTOR_OBSERVACION: VECTOR QUE CONTIENE LA INFOR-               *
 *           MACION A SER GUARDADA EN EL PROCESO                             *
 *           NUMERO_INDICE: NUMERO DEL INDICE DEL PROCESO                    *
 *           A EL CUAL SE LE VA A ANEXAR LA OBSERVACION                      *
 *           NUMERO_ARC: NUMERO DEL ARCHIVO DONDE VA A                       *
 *           SER ANEXADA LA OBSERVACION                                      *
 *           CLAVE2: CONTIENE EL NOMBRE DEL PROCESO DONDE                    *
 *           A DE SER GUARDADA LA OBSERVACION                                *
 *           CADENA_LEIDA: CADENA QUE TIENE LA INFORMACION DEL               *
 *           PROCESO                                                         *
 *           CONT_AU_PROF: PROFORMA EN MEMORIA                               *
 *           (UTIL PARA LOS DEMAS PROCESOS)                                  *
 *                                                                           *
 *      LLAMADA POR:                                                         *
 *           SELECCION                                                       *
 *                                                                           *
 *      LLAMA A:                                                             *
 *           GOTOXY                                                          *
 *           BUSCA                                                           *
 *                                                                           *
 *     ARCHIVOS:                                                             *
 *           CONTROL_ARCHIVO: CONTIENE INFORMACION DE LOS ARCHIVOS           *
 *           PRINCIPALES DEL SISTEMA                                         *
 *                                                                           *
 *****************************************************************************/

DCL
CK                                                  CHAR(1) VAR,
(LON,LONGITUD)                                       BIN FIXED,
(OUT,IN)                                          FILE,
(OR,CONT_AU_PROF,CONT_AU_FACT,CONT_AU_PED,CONTADOR_RETN,
CONT_AU_CHEQ,CONT_AU_PAG,CONT_AU_RECI)             BIN FIXED EXT STATIC,
(PROF_AUX,FACT_AUX,PAG_AUX,RECI_AUX)                BIN FIXED EXT STATIC,
(CONTADOR_PP,CONTADOR_PP,CONTADOR_AU)               BIN FIXED EXT STATIC,
(PED_AUX,CHEQ_AUX)                                  BIN FIXED EXT STATIC,
VECTOR_OBSERVACION(3)                               CHAR(70) VAR,
VALOR                                               CHAR(70) VAR,
(NUMERO_INDICE,NUMERO_ARC,I)                        BIN FIXED,
CLAVE2                                              CHAR(120) VAR,
CADENA_LEIDA                                        CHAR(1000) VAR,
(API_INV1,API_INV2)                                 DEC FIXED,
NREF                                                BIN FIXED,
CONTROL_ARCHIVO                                     FILE,
(INDICE,ARCH_DATOS,REINTEGRO)                       FILE,
DTR                                                 DEC FIXED,
TESAURO                                             FILE VARIABLE,
BA_ENC                                              BIT(1),
(POS_ARS,POS,PF_CM,LON_CM,PF_CR,UNIT_N,POS_RE,VIA(5))  BIN FIXED,
RES_CM                                              CHAR(120) VAR,
01 INF,
   02 APU_RAIZ                                      BIN FIXED,
   02 NO_NTU,                                       BIN FIXED,
   02 NO_NODO                                       BIN FIXED,
```

```
   02 APE PROX VAC(1)                                   BIN FIXED,
01 NODO,
   02 CAR_NODO,
      03 NUM_N                                          BIN FIXED,
      03 NTV                                            BIN FIXED,
      03 NC                                             BIN FIXED,
      03 TN                                             BIN FIXED,
      03 VECINA_I                                       BIN FIXED,
      03 VECINA_D                                       BIN FIXED,
   02 ENTRADA                                           CHAR(1009) VAR,
01 BLQ_NODO_PP,
   02 CADENA_LETRAS                                     CHAR(1000) VAR,
01 BLQ_PROD,
   02 CADENA_LETRA1                                     CHAR(1000) VAR,
01 BLQ_RETN,
   02 CADENA_RETN                                       CHAR(500) VAR,
   02 BLA                                               BIN FIXED,
   02 BLS                                               BIN FIXED,
   02 OBSERVACION_RETN(3)                               CHAR(70) VAR,
01 BLQ_CHEQ,
   02 CADENA_CHEQ                                       CHAR(1000) VAR,
01 BLQ_PROF,
   02 CADENA_PROF                                               CHAR(1000) VAR,
01 BLQ_FACT,
   02 CADENA_FACT                                               CHAR(1000) VAR,
01 BLQ_PED,
   02 CADENA_PED                                                CHAR(1000) VAR,
01 BLQ_PAG,
   02 CADENA_PAG                                                CHAR(1000) VAR,
01 BLQ_RECI,
   02 CADENA_RECI                                               CHAR(1000) VAR,
01 CONTROL_ARCH(1A),
   02 TIPO                                              BIN FIXED,
   02 N_ARC                                             BIN FIXED,
   02 LON_REG                                           BIN FIXED,
   02 NLT_VAC(0)                                        BIN FIXED,
   02 MAX_DIR                                           BIN FIXED,
   02 NOMBRE_ARC                                        CHAR(20) VAR,
(PROFORMA1,FACTURA1,PEDIDO1,PAGO1,RECIBIR1) CHAR(12) VAR EXT STATIC,
CHEQUE1                                                CHAR(12) VAR EXT STATIC,
(CLAVE1,AUTOR1,PP1)                                    CHAR(120) VAR EXT STATIC,
GOTOXY ENTRY(BIN FIXED,BIN FIXED),
BUSCA ENTRY(FILE VARIABLE,CHAR(120) VAR,RTT(1),BIN FIXED,CHAR(120) VAR,
BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,BIN FIXED,
(5) BIN FIXED,BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,
2 BIN FIXED,2 BIN FIXED);

READ FILE(CONTROL_ARCH(V0)) INTO(CONTROL_ARCH) KEY(1);

CK = ASCII(11);
TESAURO = ENDICE;
OB = 0;
CALL GOTOXY(0,20);

IF (CONT_AU_PROF=0 & CONT_AU_FACT=0 & CONT_AU_PAG=0 & CONT_AU_RECI=0 &
CONT_AU_CHEQ=0&CONT_AU_PED=0&CONTADOR_PR=0&CONTADOR_PP=0&CONTADOR_AU=0 &
CONTADOR_RETN = 0) THEN
      GOTO FIN;
DO T = 1 TO 3;
   CALL GOTOXY(0,14+T);
   IF T = 1 THEN
      PUT FILE(OUT) EDIT('OBSERVACION:',CK)(A,A);
```

```
    PUT FILE(ODD) EDIT(CK)(A);
END;
I = 0;
CONTINUE:
    I = I + 1;
    CALL GOTOXY(15,16+I);
    READ FILE(IN) INTO(VALOR);
    IF (I = 1 & VALOR = '') THEN
        GOTO FIN;
    VECTOR_OBSERVACION(I) = VALOR;
    IF I < 3 THEN
        GOTO CONTINUE;
IF CONTADOR_RETN ^= 0 THEN DO;
    READ FILE(REINTEGRO) INTO(REG_RETN) KEY(CONTADOR_RETN);
    DO I = 1 TO 3;
        OBSERVACION_RETN(I) = VECTOR_OBSERVACION(I);
    END;
    WRITE FILE(REINTEGRO) FROM(REG_RETN) KEYFROM(CONTADOR_RETN);
    GOTO FIN;
END;
IF CONT_AU_PROF ^= 0 THEN DO;
    NUMERO_INDICE = 13;
    NUMERO_ARC = 3;
    CLAVE2 = PROFORMA1;
    CALL BUSCA_DIRE;
    READ FILE(ARCH_DATOS) INTO(REG_PROF) KEY(DIR);
    CADENA_IMPRIMIR = CADENA_PROF;
END;
IF CONT_AU_FACT ^= 0 THEN DO;
    NUMERO_INDICE = 14;
    NUMERO_ARC = 4;
    CLAVE2 = FACTURA1;
    CALL BUSCA_DIRE;
    READ FILE(ARCH_DATOS) INTO(REG_FACT) KEY(DIR);
    CADENA_IMPRIMIR = CADENA_FACT;
END;
IF CONT_AU_PED ^= 0 THEN DO;
    NUMERO_INDICE = 15;
    NUMERO_ARC = 5;
    CLAVE2 = PEDIDO1;
    CALL BUSCA_DIRE;
    READ FILE(ARCH_DATOS) INTO(REG_PED) KEY(DIR);
    CADENA_IMPRIMIR = CADENA_PED;
END;
IF CONT_AU_CHEQ ^= 0 THEN DO;
    NUMERO_INDICE = 16;
    NUMERO_ARC = 6;
    CLAVE2 = CHEQUE1;
    CALL BUSCA_DIRE;
    READ FILE(ARCH_DATOS) INTO(REG_CHEQ) KEY(DIR);
    CADENA_IMPRIMIR = CADENA_CHEQ;
END;
IF CONT_AU_RECI ^= 0 THEN DO;
    NUMERO_INDICE = 18;
    NUMERO_ARC = 8;
    CLAVE2 = RECIBO1;
    CALL BUSCA_DIRE;
    READ FILE(ARCH_DATOS) INTO(REG_RECI) KEY(DIR);
    CADENA_IMPRIMIR = CADENA_RECI;
END;
IF CONT_AU_PAG ^= 0 THEN DO;
    NUMERO_INDICE = 17;
    NUMERO_ARC = 7;
```

```
   CLAVE2 = PAG01;
   CALL BUSCA_DIRE;
   READ FILE(ARCH_DATOS) INTO(BLQ_PAG) KEY(DIR);
   CADENA_LEIDA = CADENA_PAG;
END;
IF CONTADOR_PR ^= 0 THEN DO;
   NUMERO_INDICE = 12;
   NUMERO_ARC = 2;
   CLAVE2 = CLAVE1;
   CALL BUSCA_DIRE;
   READ FILE(ARCH_DATOS) INTO(BLQ_PROV) KEY(DIR);
   CADENA_LEIDA = CADENA_LEIDA1;
END;
IF (CONTADOR_PP ^= 0 | CONTADOR_AU ^= 0) THEN DO;
   NUMERO_INDICE = 11;
   NUMERO_ARC = 1;
   IF CONTADOR_PP ^= 0 THEN
      CLAVE2 = PP1;
   ELSE
      CLAVE2 = AUTOR1;
   CALL BUSCA_DIRE;
   READ FILE(ARCH_DATOS) INTO(BLQ_MONO_PP) KEY(DIR);
   CADENA_LEIDA = BLQ_MONO_PP.CADENA_LEIDA2;
END;
POS = LENGTH(CADENA_LEIDA);
IF POS > 450 THEN DO;
   CALL GOTOXY(0,22);
PUT FILE(OUT) EDIT("NO HAY MAS ESPACIO EN LA CADENA PARA MAS OBSERVACIONES")(A)
   PUT FILE(OUT) SKIP EDIT("CONSULTE AL ANALISTA")(A);
   GOTO FIN;
END;
LON = RANK(SUBSTR(CADENA_LEIDA,POS,1));
IF LON = 255 THEN DO;
   CADENA_LEIDA = CADENA_LEIDA || ASCII(251) ||
   ASCII(LENGTH(VECTOR_OBSERVACION(1))+1) || VECTOR_OBSERVACION(1) ||
   ASCII(LENGTH(VECTOR_OBSERVACION(2))+1) || VECTOR_OBSERVACION(2) ||
   ASCII(LENGTH(VECTOR_OBSERVACION(3))+1) || VECTOR_OBSERVACION(3) ||
   ASCII(252) || ASCII(250);
END;
ELSE DO;
   CADENA_LEIDA = SUBSTR(CADENA_LEIDA,1,POS-1) || ASCII(251) ||
   ASCII(LENGTH(VECTOR_OBSERVACION(1))+1) || VECTOR_OBSERVACION(1) ||
   ASCII(LENGTH(VECTOR_OBSERVACION(2))+1) || VECTOR_OBSERVACION(2) ||
   ASCII(LENGTH(VECTOR_OBSERVACION(3))+1) || VECTOR_OBSERVACION(3) ||
   ASCII(252) || ASCII(250);
END;
IF CONT_AU_PROF ^= 0 THEN DO;
   CADENA_PROF = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_PROF) KEYFROM(DIR);
END;
IF CONT_AU_FACT ^= 0 THEN DO;
   CADENA_FACT = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_FACT) KEYFROM(DIR);
END;
IF CONT_AU_RECI ^= 0 THEN DO;
   CADENA_RECI = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_RECI) KEYFROM(DIR);
END;
IF CONT_AU_PED ^= 0 THEN DO;
   CADENA_PED = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_PED) KEYFROM(DIR);
END;
IF CONT_AU_CHEQ ^= 0 THEN DO;
```

```
   CADENA_CHEQ = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_CHEQ) KEYFROM(DIR);
END;
IF CONT_AU_PAR ^= 0 THEN DO;
   CADENA_PAR = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_PAR) KEYFROM(DIR);
END;
IF CONTADOR_PR ^= 0 THEN DO;
   CADENA_LEIDA1 = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_PROV) KEYFROM(DIR);
END;
IF (CONTADOR_PP ^= 0 | CONTADOR_AU ^= 0) THEN DO;
   BLQ_MONO_PP.CADENA_LEIDA2 = CADENA_LEIDA;
   WRITE FILE(ARCH_DATOS) FROM(BLQ_MONO_PP) KEYFROM(DIR);
END;
BUSCA_DIRE: PROCEDURE;

   CLOSE FILE(INDICE);
   I = 1;
   DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
      I = I + 1;
   END;
   OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
   DIRECT KEYED ENV(RECSIZE(1024));
   I = 1;
   DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_ARC);
      I = I + 1;
   END;
   LONGITUD = CONTROL_ARCH(I).LON_REG;
   CLOSE FILE(ARCH_DATOS);
   OPEN FILE(ARCH_DATOS) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
   DIRECT KEYED ENV(RECSIZE(LONGITUD));
   CALL BUSCA(TESAURO,CLAVE2,RAIZ_FNC,POS_ARS,NES_CH,PF_CH,LON_CH,NODO,
   PF_CR,VIA,HIJ_N,POS_RE,INF);
   DIR = DEC(RANK(SUBSTR(ENTRADA,POS_ARS+2,1)))*10000 +
   RANK(SUBSTR(ENTRADA,POS_ARS+3,1))*100 +
   RANK(SUBSTR(ENTRADA,POS_ARS+4,1));

END;    /*      BUSCA_DIRE   */
FIN;
END;    /*       CREA_OBSERVACION      */
```

```
MUEVE_REG: PROCEDURE(J);

/*****************************************************************
 *                                                               *
 *  RUTINA QUE SE ENCARGA DE MOVERSE ENTRE LOS REGISTROS         *
 *  DEL ARCHIVO REINTEGRO                                        *
 *                                                               *
 *  PARAMETROS:                                                  *
 *                                                               *
 *     J:INDICA SI SE VA A MOSTRAR EL REINTEGRO SIGUIENTE        *
 *       O ANTERIOR AL REINTEGRO EN MEMORIA                      *
 *       1: REINTEGRO SIGUIENTE                                  *
 *       0: REINTEGRO ANTERIOR                                   *
 *                                                               *
 *  VARIABLES:                                                   *
 *                                                               *
 *     DIRE_PANTA: DIRECCION DEL REGISTRO QUE CONTIENE LA        *
 *                 PANTALLA DEL REINTEGRO EN EL ARCHIVO          *
 *                 PANTALLAS                                     *
 *                                                               *
 *     DIRE_COR: DIRECCION DEL REGISTRO QUE CONTIENE LAS         *
 *               COORDENADAS DE LOS DATOS DE LA PANTALLA         *
 *               EN EL ARCHIVO COORDENADAS                       *
 *                                                               *
 *     IG: INICIO DE LAS COORDENADAS EN EL VECTOR DE COORDENADAS *
 *                                                               *
 *     FG: FIN DE LAS COORDENAS                                  *
 *                                                               *
 *     CONTADOR_REIN: NUMERO DEL REGISTRO EN CONSULTA            *
 *                                                               *
 *  LLAMADA POR:                                                 *
 *                                                               *
 *     MAS_MENOS_LINEAS                                          *
 *                                                               *
 *  LLAMA A:                                                     *
 *                                                               *
 *     RECONSTRUYE                                               *
 *                                                               *
 *     GOTOXY                                                    *
 *                                                               *
 *  ARCHIVOS:                                                    *
 *                                                               *
 *     REINTEGROS: CONTIENE LOS REINTEGROS                       *
 *                                                               *
 *     PANTALLAS: CONTIENE TODAS LAS PANTALLAS DEL SISTEMA       *
 *                                                               *
 *     COORDENADAS: CONTIENE LAS COORDENADAS DE LOS DATOS        *
 *                  DE LA PANTALLA                               *
 *                                                               *
 *  MACRO ALGORITMO DEL PROGRAMA:                                *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
 *                                                               *
```

```
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/************************************************************************/

DCL
CADENA_LEIDA                                        CHAR(1000) VAR,
(I,C,DIRE_PANTA,DIRE_COR,IV,FV,IG,FG)               BIN FIXED,
CONTADOR_REIN                                  BIN FIXED EXT STATIC,
(IN,OUT,REINTEGRO,COORDENADAS,PANTALLAS)            FILE,
01 BLQ_REIN,
   02 CADENA_REIN                                   CHAR(500) VAR,
   02 BLA                                           BIN FIXED,
   02 BLS                                           BIN FIXED,
   02 OBSERVACION_REIN(3)                           CHAR(70) VAR,
RECONSTRUYE ENTRY(BIN FIXED,BIN FIXED,BIN FIXED,BIN FIXED,BIN FIXED,
BIN FIXED,CHAR(1000) VAR),
GOTOXY ENTRY(BIN FIXED,BIN FIXED);

DIRE_PANTA = 10;
DIRE_COR = 30;
IG = 1;
FG = 100;
IV = 1;
FV = 100;
IF J = 1 THEN DO;
   READ FILE(REINTEGRO) INTO(BLQ_REIN) KEY(CONTADOR_REIN);
   CONTADOR_REIN = CONTADOR_REIN + 1;
   IF BLS = CONTADOR_REIN THEN DO;
      READ FILE(REINTEGRO) INTO(BLQ_REIN) KEY(CONTADOR_REIN);
      CADENA_LEIDA = CADENA_REIN;
      CALL RECONSTRUYE(DIRE_PANTA,DIRE_COR,IG,FG,IV,FV,CADENA_LEIDA);
   END;
   ELSE DO;
      CONTADOR_REIN = CONTADOR_REIN - 1;
      CALL GOTOXY(0,23);
      PUT FILE(OUT) EDIT('NO EXISTEN MAS REINTEGROS')(A);
      GOTO FIN;
   END;
END;
IF J = 2 THEN DO;
   CONTADOR_REIN = CONTADOR_REIN - 1;
   IF CONTADOR_REIN = 0 THEN DO;
      CONTADOR_REIN = 1;
      CALL GOTOXY(0,23);
      PUT FILE(OUT) EDIT('NO EXISTEN MENOS REINTEGROS')(A);
      GOTO FIN;
   END;
   ELSE DO;
```

```
      READ FILE(REINTEGRO) INTO(REG_REIN) KEY(CONTADOR_REIN);
      CADENA_LEIDA = CADENA_REIN;
      CALL RECONSTRUYE(DIRE_PANTA,DIRE_COR,IG,FG,IV,FV,CADENA_LEIDA);
   END;
END;

DO I = 1 TO 3;
   CALL GOTOXY(0,18+I);
   PUT FILE(OUT) EDIT(OBSERVACION_REIN(I))(A);
END;
FIN;

END;  /*      MUEVE_REIN     */
```

```
BUSCA: PROCEDURE(TESAURO,RES_CR,BA_ENC,POS_ABS,RES_CM,PF_CM,LON_CM,
        NODO,PF_CR,VTA,ULT_N,POS_RF,ENF);

/*****************************************************************************
*                                                                            *
*      BUSCA                                                                 *
*                                                                            *
*     RUTINA QUE BUSCA UNA CLAVE EN EL ARBOL, DETERMINA SI EXIS-             *
*     TE O NO; INFORMA LA POSICION RELATIVA Y ABSOLUTA DONDE ES-             *
*     TA, EL CAMINO DE RECORRIDO DE BUSQUEDA, LA HOJA INVOLUCRA-             *
*     DA Y EL PREFIJO QUE TIENE (O DEBE TENER).                              *
*                                                                            *
*      LLAMADA POR:                                                          *
*                  INCORPORAR                                                *
*                  RECUPERAR                                                 *
*       RUTINAS QUE LLAMA:                 ------                           *
*                                                                            *
*      PARAMETROS:                                                           *
*                                                                            *
*        DE ENTRADA:                                                         *
*           TESAURO: ARCHIVO DONDE SE VA A BUSCAR                            *
*           RES_CR: CLAVE BUSCADA                                            *
*                                                                            *
*        DE SALIDA:                                                          *
*           BA_ENC: BANDERA ENCONTRADA -> 0 = NO                             *
*           POS_ABS: POSICION ABSOLUTA (Nro. DE CLAVE) DONDE FI-             *
*              NALIZO LA BUSQUEDA                                            *
*           LON_CM: LONGITUD DE RES_CM                                       *
*           AP: APUNTADOR ASOCIADO A CM                                      *
*           NODO: ESTRUCTURA DE DEL ARCHIVO ARBOL; CONTIENE EL               *
*              NODO HOJA DONDE FINALIZO LA BUSQUEDA                          *
*           PF_CR: PREFIJO DE LA CLAVE BUSCADA AL MOMENTO DE FI-             *
*              NALIZAR LA BUSQUEDA                                           *
*           VTA: VECTOR QUE CONTIENE LAS DIRECCIONES DE LOS NO               *
*              DOS INVOLUCRADOS EN LA BUSQUEDA                               *
*                                                                            *
*****************************************************************************/

/*
                  DECLARACION DE VARIABLES
                                                                          */
DCL
   POS_ABS                          BIN FIXED,
   POS_RF                           BIN FIXED,
   BA_ENC                           BIT(1),
   BA_DEC                           BIT(1),
   IGUAL                            BIT(1),
   RES_CR                           CHAR(120) VAR,
        /* RES_CR = CR A PARTIR DE PF_CR + 1  */
   PF_CR                            BIN FIXED,
   RES_CM                           CHAR(120) VAR,
   LON_REG_PAR                      BIN FIXED,
   PF_CM                            BIN FIXED,
   LON_CM                           BIN FIXED,
   AP                               DEC FIXED,
   APH                              BIN FIXED,
   VTA(5)                           BIN FIXED,
   ULT_N                            BIN FIXED,
   I                                BIN FIXED,
   1 ENF,
     2 APU_RAIZ                     BIN FIXED,
     2 NO_NIV                       BIN FIXED,
     2 NO_NRO                       BIN FIXED,
```

```
      2 APU_PRUX_VACIO              BIN FIXED,
    1 NODO,
      2 CAR_NODO,
        3 NUM_N                     BIN FIXED,
        3 NIV                       BIN FIXED,
        3 NC                        BIN FIXED,
        3 IN                        BIN FIXED,
        3 VECINA_I                  BIN FIXED,
        3 VECINA_D                  BIN FIXED,
      2 ENTRADA                     CHAR(1009) VAR,
    TESAURO                         FILE VARIABLE;
/*
                    PROGRAMA   PRINCIPAL
                                                                    */
READ FILE (TESAURO) INTO (INF) KEY (1);
READ FILE (TESAURO) INTO (NODO) KEY (APU_RAIZ);
NUM_N = 1;
VIA(NUM_N) = APU_RAIZ;
BA_ENC = '0'B;
BA_DEC = '0'B;
POS_ABS = 1;
POS_RE = 0;
PF_CB = 0;
LON_CM = 0;
IF NIV ^= 1 THEN LON_REG_PAR = 4;
ELSE LON_REG_PAR = 10;

/*  Lazo de repeticion de busqueda  */

REPETA:
   POS_RE = POS_RE + 1;
   POS_ABS = POS_ABS + LON_CM;
   IF POS_RE = NC+1 THEN RETURN;
   PF_CM = RANK(SUBSTR(ENTRADA,POS_ABS,1));
   LON_CM = RANK(SUBSTR(ENTRADA,POS_ABS+1,1));
   IF PF_CM = PF_CB THEN DO;
      RES_CM = SUBSTR(ENTRADA,POS_ABS + LON_REG_PAR,LON_CM - LON_REG_PAR);
      IF SUBSTR(RES_CB,PF_CB+1) > RES_CM THEN DO;
        /*  Si la CB > CM -> actualice el PF_CB  */
         I = 0;
      S;  I = I + 1;
         IGUAL = '0'B;
         IF SUBSTR(RES_CB,PF_CB+I,1) = SUBSTR(RES_CM,I,1) THEN DO;
            IGUAL = '1'B;
            IF I < LON_CM-LON_REG_PAR   THEN GOTO S;
         END;
         IF I = LON_CM-LON_REG_PAR & IGUAL   THEN
            PF_CB = PF_CB + 1;
         ELSE
            PF_CB = PF_CB + I - 1;
      END;

/*  Si CB <= CM  -> Lea la hija si NIV ^= 1, de lo contrario
      determina si es igual  */

      ELSE DO;
         IF NIV ^= 1 THEN  CALL LEE_HIJO;
         ELSE DO;
            BA_DEC = '1'B;
            IF SUBSTR(RES_CB,PF_CB+1) = RES_CM THEN
               BA_ENC = '1'B;
            ELSE
               BA_ENC = '0'B;
```

```
            END;
        END;
    END;
    ELSE DO;
        IF PF_CR > PF_CM THEN DO;
            IF NIV = 0 THEN CALL LEE_HIJO;
            ELSE DO;
                BA_DEC = '1'B;
                BA_FNC = '0'B;
            END;
        END;
    END;
IF ^(BA_DEC) THEN GOTO REPITA;

LEE_HIJO:PROCEDURE;
    AP = RANK(SUBSTR(ENTRADA,POS_ABS+2,1))*256 +
         RANK(SUBSTR(ENTRADA,POS_ABS+3,1));
    READ FILE (DIESAURO) INTO (NODO) KEY (AP);
    IF NIV = 1 THEN DON_REG_PAR = 0;
    UBI_N = UBI_N + 1;
    VTA(UBI_N) = AP;
    PF_CR = 0;
    POS_ABS = 1;
    POS_RF = 0;
    DON_CM = 0;
END; /*  Fin de LEE_HIJO  */

END; /*  Fin de BUSCA  */
```

```
PTDE_VACIO: PROCEDURE(NOMBRE_ARCHIVO,DIRECCION);

/*****************************************************************************
 *                                                                           *
 *    RUTINA QUE SE ENCARGA DE SUMINISTRAR LOS BLOQUES VACIOS                *
 *    PARA EL ASENTAMIENTO DE INFORMACION EN LOS ARCHIVOS                    *
 *    PRINCIPALES DEL SISTEMA                                                *
 *                                                                           *
 *       PARAMETROS:                                                         *
 *                                                                           *
 *            NOMBRE_ARCHIVO: NOMBRE DEL ARCHIVO A EL CUAL SE LE             *
 *                            VA A SUMINISTRAR EL BLOQUE                     *
 *                                                                           *
 *            DIRECCION: DIRECCION DE BLOQUE SUMINISTRADO                    *
 *                                                                           *
 *       VARIABLES:                                                          *
 *                                                                           *
 *       LLAMADA POR:                                                        *
 *            FINALIZA                                                       *
 *                                                                           *
 *       LLAMA A:                                                            *
 *                                                                           *
 *       ARCHIVOS:                                                           *
 *                                                                           *
 *            CONTROL_ARCHIVOS: CONTIENE INFORMACION DE LOS                  *
 *                              PRINCIPALES ARCHIVOS DEL SISTEMA             *
 *                                                                           *
 *****************************************************************************/


DCL
OUT                                                     FILE,
CONTROL_ARCHIVO                                         FILE,
I                                                       BIN FIXED,
DIRECCION                                               DEC FIXED,
NOMBRE_ARCHIVO                                          CHAR(20) VAR;
DCL
01 CONTROL_ARCH(16),
   02 TIPO                                              CHAR(1),
   02 N_ARC                                             BIN FIXED,
   02 CON_REG                                           BIN FIXED,
   02 ULT_VACIO                                         BIN FIXED,
   02 MAX_DIR                                           BIN FIXED,
   02 NOMBRE_ARC                                        CHAR(20) VAR;

READ FILE(CONTROL_ARCHIVO) INTO(CONTROL_ARCH) KEY(1);

I = 1;
DO WHILE(NOMBRE_ARCHIVO ^= NOMBRE_ARC(I));
   I = I + 1;
END;
IF ULT_VACIO(I) = 0 THEN DO;
   MAX_DIR(I) = MAX_DIR(I) + 1;
   DIRECCION = MAX_DIR(I);
END;

WRITE FILE(CONTROL_ARCHIVO) FROM(CONTROL_ARCH) KEYFROM(1);
END; /*              PTDE_VACIO              */
```

```
/*****************************************************************************
*                                                                            *
*    GOTOXY:                                                                 *
*           Rutina encargada de posicionarse en la fila y columna dada.      *
*                                                                            *
*    Parametros:  COL:  columna en que se debe posicionar  (0 - 79)          *
*          FILA: fila en que se debe posicionar. (0 - 23)                    *
*                                                                            *
*    Llamada por:  PRINCIPAL, GENERA, LEE_MODIF, INCORPORAR, MODIFICAR       *
*                  INFORMAR, LEE_RESTO_COMANDO, VALIDA_CLAVE,  LISTAR,       *
*                  GUARDA_INF, RECONSTRUYE.                                  *
*                                                                            *
*    Llama a:                                                                *
*                                                                            *
*    Variables:  CP:  CTRL P     variables especiales para el manejo en      *
*                CX:  CTRL X     de pantalla de la DATA GENERAL.             *
*                CZ   CTRL Z                                                 *
*                                                                            *
*    Archivos:  CONSOLA: pantalla                                            *
*                                                                            *
*****************************************************************************/

GOTOXY:  PROCEDURE(COL,FILA);

DCL      (COL,FILA)                 BIN FIXED,
         (CP,CX,CZ)                 CHAR(1);
DCL      OUT                        FILE;

IF ( FILA>23 | FILA<0 )  THEN FILA = 0;
IF ( COL >79 | COL <0 )  THEN COL  = 0;
CP = ASCII(16);
CX = ASCII(24);
CZ = ASCII(26);
PUT FILE(OUT) SKIP(0);
IF ( COL=9 & FILA^=9) THEN
     PUT FILE(OUT) EDIT(CP,ASCII(8),ASCII(FILA),CX)(A);
     ELSE IF ( COL^=9 & FILA=9) THEN
               PUT FILE(OUT) EDIT(CP,ASCII(COL),ASCII(8),CZ)(A);
               ELSE IF ( COL=9 & FILA=9 ) THEN
                    PUT FILE(OUT) EDIT(CP,ASCII(8),ASCII(8),CX,CZ)(A);
                    ELSE
                     PUT FILE(OUT) EDIT(CP,ASCII(COL),ASCII(FILA))(A);
END;    /*   GOTOXY   */
```

```
%PROCESS; PROCEDURE(DIRE_PANTA,AUXILIAR,CONT_AU_ACEP,
VECTOR_GUAR_NUM,VECTOR_GUAR_AP2,NUMERO_ACTIVIDAD,
DIR,NTITUL,NEJEMP,ARCH_DATOS,CONT_AU_ACEP_AUX,PRE);
```

```
/*****************************************************************************
 *                                                                           *
 *     PROGRAMA ENCARGADO DE EVITAR QUE UN AUTOR O UNA PUBLICACION           *
 *     SEA INCLUIDO EN UN PROCESO                                            *
 *                                                                           *
 *     PARAMETROS:                                                           *
 *     DIRE_PANTA: DIRECCION DEL REGISTRO CORRESPONDIENTE A LA               *
 *     PANTALLA DEL PROCESO                                                  *
 *     AUXILIAR: POSICION DEL AUTOR/PP EN EL VECTOR GUAR_NUM                 *
 *     CONT_AU_ACEP: CONTADOR DE AUTORES ACEPTADOS                           *
 *     VECTOR_GUAR_NUM: VECTOR QUE GUARDA LOS NUMEROS QUE                    *
 *     CORRESPONDEN A LOS AUTORES Y/O PUBLICACIONES A SER                    *
 *     INCLUIDAS EN EL PROCESO                                               *
 *     VECTOR_GUAR_AP2: VECTOR QUE GUARDA LOS INDICADORES DE                 *
 *     AUTORES Y/O PP:                                                       *
 *     1: AUTOR                                                              *
 *     0: PP                                                                 *
 *     NUMERO_ACTIVIDAD: NUMERO DE LA ACTIVIDAD QUE SE ESTA                  *
 *     CREANDO                                                               *
 *     DIR: DIRECCION DEL REGISTRO DONDE SE VA A GUARDAR LA                  *
 *     INFORMACION DEL PROCESO                                               *
 *     NTITUL: NUMERO DE TITULOS                                             *
 *     NEJEMP: NUMERO DE EJEMPLARES                                          *
 *                                                                           *
 *     LLAMADO POR:                                                          *
 *     MUESTRA_INFORMACION                                                   *
 *                                                                           *
 *     LLAMA A:                                                              *
 *                                                                           *
 *     VARIABLES:                                                            *
 *     REG_MUND_PP: BLOQUE DE INFORMACION DE AUTOR O PP                      *
 *                                                                           *
 *     ARCHIVOS:                                                             *
 *     ARCH_DATOS: ARCHIVO DE DATOS                                          *
 *                                                                           *
 *****************************************************************************/
```

```
DCL
(RAN,PRE)                                                  BIT(1),
(DIRE_PANTA,NUMERO_INDICE,I,POS1,CONT,POS2)                BIN FIXED,
(ARCH_DATOS,DIR)                                           FILE,
(AUXILIAR,CONT_AU_ACEP,CONT_AU_ACEP_AUX)                   BIN FIXED,
VECTOR_GUAR_NUM(10)                                        BIN FIXED,
VECTOR_GUAR_AP2(10)                                        DEC FIXED,
(NUMERO_ACTIVIDAD,NUMERO_ACTIVIDAD_AUX)                    CHAR(12) VAR,
 NUMERO_ACTIVIDAD (                                        CHAR(12) VAR,
DIR                                                        DEC FIXED,
01 REG_MUND_PP,
   02 CADENA_ENTDA2                                        CHAR(1000) VAR,
(NEJEMP,NTITUL,NEJEMP_IE)                                  BIN FIXED;

READ FILE(ARCH_DATOS) INTO(REG_MUND_PP) KEY(DIR);
RAN = "0"B;

        IF AUXILIAR < CONT_AU_ACEP THEN DO;
           DO WHILE(AUXILIAR < CONT_AU_ACEP);
              VECTOR_GUAR_NUM(AUXILIAR) = VECTOR_GUAR_NUM(AUXILIAR+1);
              VECTOR_GUAR_AP2(AUXILIAR) = VECTOR_GUAR_AP2(AUXILIAR+1);
              AUXILIAR = AUXILIAR + 1;
```

```
            END;
            RAN = "1"B;
         END;
         IF RAN = "0"B THEN
            PRT = "0";
         CONT_AU_ACEP = CONT_AU_ACEP - 1;
         CONT_AU_ACEP_AUX = CONT_AU_ACEP_AUX - 1;
         AUXILIAR = AUXILIAR - 1;
         IF CONT_AU_ACEP = 0 THEN
            AUXILIAR = AUXILIAR + 1;
         CALL BUSCA_POSICION;
         POS2 = POS1;
         IF DIRE_PANTA = 4 THEN DO;
            POS1 = POS1 + LON1;
            LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
            NEJEMP_IF = BINARY(SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1));
            NECTRL = NECTRL - 1;
            NEJEMP = NEJEMP - NEJEMP_IF;
         END;
         POS1 = 1;
         LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
         DO WHILE(LON1 ^= 255);
            POS1 = POS1 + LON1;
            LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
         END;
CADENA_LEIDA2 = SUBSTR(CADENA_LEIDA2,1,POS2-1)||SUBSTR(CADENA_LEIDA2,POS1);
         WRITE FILE(ARCH_DATOS) FROM(REG_NOND_PP) KEYFROM(DIR);

BUSCA_POSICION: PROCEDURE;

         IF DIRE_PANTA = 4 THEN
            NUMERO_ACTIVIDAD_AUX = "PF"||NUMERO_ACTIVIDAD;
         IF DIRE_PANTA = 5 THEN
            NUMERO_ACTIVIDAD_AUX = "FA"||NUMERO_ACTIVIDAD;
         IF DIRE_PANTA = 6 THEN
            NUMERO_ACTIVIDAD_AUX = "PD"||NUMERO_ACTIVIDAD;
         IF DIRE_PANTA = 8 THEN
            NUMERO_ACTIVIDAD_AUX = "PA"||NUMERO_ACTIVIDAD;
         IF DIRE_PANTA = 9 THEN DO;
            NUMERO_ACTIVIDAD_1 = NUMERO_ACTIVIDAD;
            DO I = 1 TO 9 - LENGTH(NUMERO_ACTIVIDAD_1);
               NUMERO_ACTIVIDAD_1 = " " || NUMERO_ACTIVIDAD_1;
            END;
            NUMERO_ACTIVIDAD_AUX = "RF"||NUMERO_ACTIVIDAD_1;
         END;
         POS1 = 1;
         LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
   DO WHILE(NUMERO_ACTIVIDAD_AUX ^= SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1));
            POS1 = POS1 + LON1;
            LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
      END;

END; /* BUSCA POSICION    */
FF:
END;    /*    RECHAZADO      */
```

```
ROMPER_NODO : PROCEDURE(TESAURO,NODO,VTA,RES_CM,MEN,AUX),QUE(G,RA_FIN),
                        LON_REG_PAR,ULT_N);
/*****************************************************************
*                                                                *
*     ROMPER NODO                                                *
*                                                                *
*     RUTINA QUE PARA ROMPER UN NODO CUANDO SU TAMA\O ES         *
*     MAYOR QUE EL MAXIMO                                        *
*                                                                *
*     RUTINAS QUE LLAMA:  -----                                  *
*                                                                *
*     LLAMADA POR:     INSERTA                                   *
*                                                                *
*     PARAMETROS:                                                *
*       DE ENTRADA:                                              *
*         TESAURO: ARCHIVO TRATADO                               *
*         NODO: ESTRUCTURA QUE CONTIENE INFORMACION SOBRE EL     *
*               NUMERO, NIVEL DENTRO DEL TESAURO, NUMERO DE CLA-*
*               VES(NC), TAMA\O DEL NODO(TN), NODO VECINO IZQ.,  *
*               SU NODO VECINO DER., Y FINALMENTE LAS CLAVES Y   *
*               LOS APUNTADORES(ENTRADA)                         *
*         VTA: VECTOR QUE CONTIENE LAS DIRECCIONES DE LOS NODOS  *
*              INVOLUCRADOS EN LA BUSQUEDA                       *
*                                                                *
*****************************************************************/

DCL
    (AUX1,AUX2,TN_AUX,NC_AUX)              BIN FIXED,
    (I,J,K,POS)                            BIN FIXED,
    LON_REG_PAR                            BIN FIXED,
    SUM                                    BIN FIXED,
    NC2                                    BIN FIXED,
    (VLON(5),VPF(5))                       BIN FIXED,
    (PROX_N,NUEVO_N)                       BIN FIXED,
    VTA(5)                                 BIN FIXED,
    (POS_ANS,POS_PF,LON_CM,PF_CB)          BIN FIXED,
    (PF_CM,ULT_N,LON_CB)                   BIN FIXED,
    CLAVE_PROM                             CHAR(30) VAR,
    RES_CB                                 CHAR (120)VAR,
    RES_CM                                 CHAR (120)VAR,
    (MEN,LAR)                              BIN FIXED,
    CLANRE                                 CHAR(120)VAR,
    VCAR(5)                                CHAR(120)VAR,
    SEP(4)                                 CHAR(30) VAR,
    APU                                    CHAR(8) VAR,
    RA_DEC                                 BIT(1),
    RA_FIN                                 BIT(1),
    1 INF,
       2 APU_RAIZ                          BIN FIXED,
       2 NO_NIV                            BIN FIXED,
       2 NO_DOS                            BIN FIXED,
       2 APU_PROX_VACIO                    BIN FIXED,
    1 NODO,
       2 CAR_NODO,
          3 NUM_N                          BIN FIXED,
          3 NIV                            BIN FIXED,
          3 NC                             BIN FIXED,
          3 TN                             BIN FIXED,
          3 VECINA_I                       BIN FIXED,
          3 VECINA_D                       BIN FIXED,
       2 ENTRADA                           CHAR(1009) VAR,
    TESAURO                                FILE VARIABLE;
```

```
      VECINA_I = NUEVO_N;
      REWRITE FILE (TESAURO) FROM (NODO) KEY (AUXD);
    END;

/*  SI EL NODO QUE SE ROMPIO ES LA RAIZ, CREA UNA NUEVA CON EL SEPARADOR
    Y UNA CLAVE MAXIMA                                                  */

   IF NIV = 1 THEN
      CLAVE_PROM = SEP(POS);
   ELSE
      CLAVE_PROM = VCAR(POS);
   IF DEL_N = 1 - 0  THEN DO;
      RAIZ = '1'B;
      CALL PIDE_VACIO;
      NUM_N = PROX_N;
      NIV = NIV + 1;
      NC = 2;
      IN = MEN + 9;
      VECINA_I = 0;
      VECINA_D = 0;
      APU_RAIZ = NUM_N;
      NO_NIV = NO_NIV + 1;
      ENTRADA = ASCII(0)!!ASCII(REN+4)!!ASCII(AUXI/256)!!
        ASCII(AUXI)!!CLAVE_PROM!!ASCII(0)!!ASCII(0)!!ASCII(NUEVO_N/256)
        !!ASCII(NUEVO_N)!!ASCII(255);
      WRITE FILE (TESAURO) FROM (NODO) KEYFROM (NUM_N);
      REWRITE FILE (TESAURO) FROM (INF) KEY (1);
   END;

/*  SI EL NODO QUE SE ROMPIO TIENE MADRE, INSERTA EN ELLA EL SEPARADOR
    Y ACTUALIZA EL APUNTADOR DEL SEPARADOR SIGUIENTE AL INSERTADO   */

   ELSE DO;
      DEL_N = DEL_N - 1;
      READ FILE (TESAURO) INTO (NODO) KEY (VIA(DEL_N));
      RES_CR = CLAVE_PROM;
      RAIZ = '0'B;
      ION_REG_PAR = 4;
      RETURN;
   END;


/*****************************************************************
          RUTINA QUE PIDE UN BLOQUE VACIO DE LOS TESAUROS
*****************************************************************/
PIDE_VACIO: PROCEDURE;

READ FILE (TESAURO) INTO (INF) KEY (1);
IF APU_PROX_VACIO = 0 THEN DO;
   NO_NOD = NO_NOD + 1;
   PROX_N = NO_NOD;
END;
ELSE DO;
   READ FILE (TESAURO) INTO (NODO) KEY (APU_PROX_VACIO);
   PROX_N = APU_PROX_VACIO;
   APU_PROX_VACIO = NUM_N;
   NUM_N = PROX_N;
END;
REWRITE FILE (TESAURO) FROM (INF) KEY (1);
RETURN;
END; /*  FIN DE PIDE_VACIO  */
```

```
AYUDA: PROCEDURE;

/*****************************************************************
 *                                                               *
 *  MUESTRA POR PANTALLA LA EXPLICACION DEL MANEJO DE LOS        *
 *  PRINCIPALES COMANDOS DEL SISTEMA                             *
 *                                                               *
 *  PARAMETROS:                                                  *
 *                                                               *
 *  VARIABLES:                                                   *
 *                                                               *
 *  LLAMADA POR:                                                 *
 *                                                               *
 *      SELECCION                                                *
 *                                                               *
 *  LLAMA A:                                                     *
 *                                                               *
 *  ARCHIVOS:                                                    *
 *                                                               *
 *      ARCHIVO HELP                                             *
 *                                                               *
 *****************************************************************/


DCL
(CTN,OUT,ARCHIVO_HELP)                                  FILE,
 I                                                      BIN FIXED,
01 BLQ_HELP,
   02 BLA                                               BIN FIXED,
   02 BLS                                               BIN FIXED,
   02 HELP(20)                                          CHAR(80) VAR,
GOTOXY ENTRY(BIN FIXED,BIN FIXED);

OPEN FILE(ARCHIVO_HELP) RECORD INPUT DIRECT KEYED
ENV(RECSIZE(1500));

READ FILE(ARCHIVO_HELP) INTO(BLQ_HELP) KEY(1);

PUT FILE(OUT) PAGE;
DO I = 1 TO 20;
   CALL GOTOXY(4,0);
   PUT FILE(OUT) EDIT(HELP(I))(A);
END;

CLOSE FILE(ARCHIVO_HELP);

END;    /*     AYUDA    */
```

```
CREA_REINTEGRO: PROCEDURE(J);

/*****************************************************************************
*     RUTINA QUE SE ENCARGA DE LA CREACION DE LOS REINTEGROS                 *
*                                                                            *
*        PARAMATROS:                                                         *
*             J: INDICA SI EL PROCESO VA A SER INTERUMPIDO                   *
*                0: INTERUMPE EL PROCESO                                     *
*                1: NO INTERUMPE LA CREACION                                 *
*                                                                            *
*        VARIABLES: DIRE_PANTA: DIRECCION DE LA PANTALLA                     *
*                   DE CREACION DE REINTEGROS EN EL                          *
*                   ARCHIVO PANTALLAS                                        *
*                   DIRE_COR: DIRECCION DONDE SE ENCUENTRAN                  *
*                   GUARDADAS LAS COORDENAS PARA LA PANTALLA                 *
*                   DEL REINTEGRO EN EL ARCHIVO COORDENADAS                  *
*                   IG: COMIENZO EN EL VECTOR DE COORDENADAS                 *
*                   FG: FIN EN EL VECTOR DE COORDENADAS                      *
*                   CONTADOR_REIN: INDICA QUE EL REINTEGRO                   *
*                   ESTA EN MEMORIA                                          *
*                                                                            *
*        LLAMADA POR:                                                        *
*                   CREA_ACTITUDAD                                           *
*                                                                            *
*        LLAMA A:                                                            *
*                   GENERA                                                   *
*                   BUSCA_MONEDA                                             *
*                   GOTOXY                                                   *
*                   CREA_INFORMACION                                         *
*                                                                            *
*        ARCHIVOS:                                                           *
*                 COORDENADAS: CONTIENE INFORMACION DE LAS                   *
*                 COORDENADAS DE LOS DATOS DE LA PANTALLAS                   *
*                 PANTALLAS: CONTIENE LAS PANTALLAS DEL                      *
*                 SISTEMA                                                    *
*                 REINTEGROS: GUARDA LOS REINTEGROS                          *
*                                                                            *
*       MACRO ALGORITMO DEL PROGRAMA:                                        *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*****************************************************************************/

DCL
```

```
MONE                                                    CHAR(5) VAR,
ENCONTRADO                                              BIT(1),
(OUT,IN,RETINTEGRO,PANTALLAS,COORDENADAS)                 FILE,
(CONTADOR_FF,TITULO_AUX,CONT_AU_FCT)                    BIN FIXED EXT STATIC,
 CONTADOR_PR                                            BIN FIXED EXT STATIC,
(CONT_AU_PROF,CONT_AU_FACT,CONT_AU_PAG,
 CONT_AU_RECI,CONT_AU_CHEQ,CHEQ_AUX,
 PROF_AUX,FACT_AUX,PAG_AUX,RECI_AUX,
 CONT_AU_PED,PED_AUX,OR,CONTADOR_RETN) BIN FIXED EXT STATIC,
(CONTADOR_AU,CONTADOR_PP)  BIN FIXED EXT STATIC,
(DTR,DIR1,I,J)                                            BIN FIXED,
01 RET_RETN,
   02 CADENA_RETN                                         CHAR(500) VAR,
   02 RET_A                                               BIN FIXED,
   02 RET_S                                               BIT FIXED,
   02 DATAENVACTION_RETN(3)                               CHAR(20) VAR,
01 CONTROL_RETN,
   02 MAX_DIR                                             BIN FIXED,
CADENA_LEIDA                                              CHAR(1000) VAR,
(IV,FV,IG,FG,DIRE_PANTA,DIRE_COR)                         BIN FIXED,
GENERA_ENTRY(BIN FIXED,BIN FIXED,BIN FIXED),
BUSCA_MONEDA_ENTRY(CHAR(5) VAR,BIT(1)),
GOTOXY_ENTRY(BIN FIXED,BIN FIXED),
CREA_INFORMACION_ENTRY(BIN FIXED,BIN FIXED,BIN FIXED,BIN FIXED,BIN FIXED,BIN FI
XED,CHAR(1000) VAR);

CONTADOR_RETN = 0;
CONT_AU_CHEQ = 0;
CONT_AU_PED = 0;
PED_AUX = 1;
CHEQ_AUX = 1;
CONTADOR_PR = 0;
CONTADOR_PP = 0;
CONTADOR_AU = 0;
CONT_AU_PROF = 0;
CONT_AU_FACT = 0;
CONT_AU_PAG = 0;
CONT_AU_RECI = 0;
OR = 0;
PROF_AUX = 1;
FACT_AUX = 1;
RECI_AUX = 1;
PAG_AUX = 1;
DIRE_PANTA = 10;
DIRE_COR = 16;
IG = 1;
IV = 1;
FV = 100;
FG = 100;
PUT FILE(OUT) PAGE;
CALL GENERA(DIRE_PANTA,IG,FG);
CALL CREA_INFORMACION(DIRE_PANTA,DIRE_COR,IG,FG,IV,FV,CADENA_LEIDA);
IF FV = 0 THEN DO;
   I = 0;
   GOTO FIN;
END;
LL:
CALL GOTOXY(35,3);
READ FILE(IN) INTO(MONE);
CALL BUSCA_MONEDA(MONE,ENCONTRADO);
IF ENCONTRADO = '0'B THEN
   GOTO LL;
CADENA_LEIDA = CADENA_LEIDA || ASCII(LENGTH(MONE)+1) || MONE || ASCII(250);
DO I = 1 TO 3;
```

```
    OBSERVACION_REIN(I) = ' ';
END;
CALL PIDE_VACIO_REIN;
IF DIR = 1 THEN DO;
    BLA = 0;
    BLS = 0;
    CADENA_REIN = CADENA_LEIDA;
    WRITE FILE(REINTEGRO) FROM(BLQ_REIN) KEYFROM(DIR);
    GOTO FIN;
END;
READ FILE(REINTEGRO) INTO(BLQ_REIN) KEY(1);
DIR1 = 1;
DO WHILE(BLS ^= 0);
    DIR1 = BLS;
    READ FILE(REINTEGRO) INTO(BLQ_REIN) KEY(BLS);
END;
BLS = DIR;
WRITE FILE(REINTEGRO) FROM(BLQ_REIN) KEYFROM(DIR1);
BLA = DIR1;
BLS = 0;
CADENA_REIN = CADENA_LEIDA;
WRITE FILE(REINTEGRO) FROM(BLQ_REIN) KEYFROM(DIR);

PIDE_VACIO_REIN: PROCEDURE;

READ FILE(REINTEGRO) INTO(CONTROL_REIN) KEY(0);
    MAX_DIR = MAX_DIR + 1;
    DIR = MAX_DIR;
WRITE FILE(REINTEGRO) FROM(CONTROL_REIN) KEYFROM(0);

END; /*     PIDE_VACIO_REIN    */
FIN:
END;    /*     CREA_REINTEGRO    */
```

```
CREA_ACTIVIDAD: PROCEDURE();

/*****************************************************************************
*                                                                            *
*      RUTINA ENCARGADA DE LA CREACION DE LOS PROCESOS DE                    *
*      PROFORMA,FACTURA,PEDIDO,PAGO Y RECIBIR                                *
*                                                                            *
*           PARAMETROS:                                                      *
*                                                                            *
*                I: INDICA EL PROCESO A SER CREADO                           *
*                                                                            *
*           VARIABLES:                                                       *
*                                                                            *
*                NUMERO_ARC: NUMERO DEL ARCHIVO DEL PROCESO                  *
*                            A CREAR                                         *
*                                                                            *
*                NUMERO_INDICE: NUMERO DEL INDICE DEL PROCESO                *
*                               A CREAR                                      *
*                                                                            *
*                DIRE_PANTA: DIRECCION DE LA PANTALLA DEL PROCESO            *
*                                                                            *
*                DIRE_COR: DIRECCION DE LAS COORDENADAS                      *
*                                                                            *
*           LLAMADA POR:                                                     *
*                                                                            *
*                SELECCION                                                   *
*                                                                            *
*           LLAMA A:                                                         *
*                                                                            *
*                GENERA_CAPTA                                                *
*                                                                            *
*           ARCHIVOS:                                                        *
*                                                                            *
*                COORDENADAS: CONTIENE LAS COORDENADAS DE LOS DATOS          *
*                             DE LAS PANTALLAS                               *
*                                                                            *
*                PANTALLAS: CONTIENE LAS PANTALLA DEL SISTEMA                *
*                                                                            *
*                CONTROL_PANTALLAS: CONTIENE LA DIRECCION DE LAS             *
*                                   PANTALLAS                                *
*                                                                            *
*****************************************************************************/

DCL
(CONT_AU_PROF,CONT_AU_FACT,CONT_AU_PAG,
 CONT_AU_RECT,CONT_AU_CHEQ,CHEQ_AUX,CONTADOR_TT,
 PROF_AUX,FACT_AUX,PAG_AUX,RECT_AUX,TTULO_AUX,
 CONT_AU_PED,PED_AUX,DR,CONTADOR_REN) BIN FIXED EXT STATIC,
(CONTADOR_PR,CONTADOR_PP,CONTADOR_AU)  BIN FIXED EXT STATIC,
(I,J,DIRE_PANTA,DIRE_COR,TR,FD,TV,FV,NUMERO_ARC,
NUMERO_INDICE)                                          BIN FIXED,
(CONT,CONTROL_PANTALLAS,COORDENADAS,PANTALLAS)          FILE,
GENERA_CAPTA ENTRY(BIN FIXED,BIN FIXED,BIN FIXED,BIN FIXED,
BIN FIXED,BIN FIXED,CHAR(1000) VAR),
CADENA_LEIDA                                            CHAR(1000) VAR,
01 CONTROL_PANTALLA,
   02 NUM_ARC                                           BIN FIXED,
   02 NUM_INDICE                                        BIN FIXED,
   02 DIRE_P                                            BIN FIXED,
   02 DIRE_C1                                           BIN FIXED,
   02 DIRE_C2                                           BIN FIXED,
   02 NOMBRE_P                                          CHAR(4) VAR,
01 REG_LEC,
```

```
    02 VECTOR COOR(500)                                    CHAR(1);

READ FILE(CONTROL_PANTALLAS) INTO(CONTROL_PANTA) KEY(1);

NR = 0;
CONTADOR_RETN = 0;
CONT_AU_PROF = 0;
CONTADOR_TIT = 0;
TITULO_AUX = 1;
CONT_AU_PED = 0;
CONTADOR_PR = 0;
CONTADOR_PP = 0;
CONTADOR_AU = 0;
PED_AUX = 1;
CONT_AU_CHEQ = 0;
CHEQ_AUX = 1;
CONT_AU_FACT = 0;
CONT_AU_PAG = 0;
CONT_AU_RECT = 0;
PROF_AUX = 1;
FACT_AUX = 1;
PAG_AUX = 1;
RECT_AUX = 1;

IF J = 20 THEN
   NUMERO_ARC = 6;
IF J = 21 THEN
   NUMERO_ARC = 4;
IF J = 34 THEN
   NUMERO_ARC = 7;
IF J = 36 THEN
   NUMERO_ARC = 5;
IF J = 41 THEN
   NUMERO_ARC = 9;
IF J = 50 THEN
   NUMERO_ARC = 3;
IF J = 52 THEN
   NUMERO_ARC = 8;
I = 1;
DO WHILE(CONTROL_PANTA(I).NUM_ARC ^= NUMERO_ARC);
   I = I + 1;
END;
DIRF_PANTA = CONTROL_PANTA(I).DIRF_P;
FG = 100;
NUMERO_INDICE = CONTROL_PANTA(I).NUM_INDICE;
DIRF_COR = CONTROL_PANTA(I).DIRE_CI;
IG = 1;
IV = 1;
FV = 100;
CALL GENERA_CAPTA(DIRF_PANTA,DIRF_COR,IG,FG,IV,FV,CADENA_ENTRADA);

END;    /*      CREA_ACTIVIDAD      */
```

```
FACTURA: PROC(NUMERO_ACTIVIDAD,CLAVE,CADENA_LEIDA);

/*****************************************************************************
*                                                                            *
*    PROGRAMA QUE REALIZA AJUSTE(CUADRE) EN LA FACTURA EN                    *
*    FORMA AUTOMATICA                                                        *
*                                                                            *
*       PARAMETROS:                                                          *
*                                                                            *
*            NUMERO_ACTIVIDAD: NUMERO DE LA FACTURA A LA CUAL                *
*                              SE LE ESTA REALIZANDO EL CUADRE               *
*            CADENA_LEIDA: CADENA QUE CONTIENE LA INFORMACION                *
*                          DE LA FACTURA                                     *
*                                                                            *
*       VARIABLES:                                                           *
*                                                                            *
*            NUMERO_INDICE: NUMERO DEL INDICE DE LA FACTURA                  *
*                                                                            *
*            NUMERO_ARC: NUMERO DEL ARCHIVO DE LA FACTURA                    *
*                                                                            *
*            VECTOR_GUAR_NUM: VECTOR QUE CONTIENE LOS NUMEROS                *
*                             DE LOS AUTORES/PP INCLUIDOS EN                 *
*                             LA FACTURA                                     *
*                                                                            *
*            VECTOR_GUAR_AP2: CONTIENE EL INDICADOR DE AUTOR                 *
*                             O PP                                           *
*                                                                            *
*            MONTO_BS: MONTO TOTAL EN BOLIVARES                              *
*                                                                            *
*            FLETE_BS: FLETE TOTAL EN BOLIVARES                              *
*                                                                            *
*            MONT_PAR: MONTO POR AUTOR/PP                                    *
*                                                                            *
*            FLETE_PAR: FLETE POR AUTOR/PP                                   *
*                                                                            *
*                                                                            *
*       LLAMADA POR:                                                         *
*                                                                            *
*            FINALIZA                                                        *
*                                                                            *
*       LLAMA A:                                                             *
*            NOTOXY                                                          *
*            PTOF_VACIO                                                      *
*            INSERTA                                                         *
*            BUSCA                                                           *
*                                                                            *
*****************************************************************************/

DCL
RA_FNC                                                  BIT(1),
NUMARR_ARCHIVO                                          CHAR(20) VAR,
FACTURA1                                    CHAR(12) VAR EXT STATIC,
(NUM_AUT,LON,LONGE(10))                                 BIN FIXED,
(MONTO_BS,FLETE_BS,MONT_PAR,FLETE_PAR)                  DEC FIXED(10,2),
(CLAVE,CLAVE2)                                          CHAR(120) VAR,
(NUMERO_INDICE,NUMERO_ARC,I,J,CONT_AU)                  BIN FIXED,
RESAUX1                                                 FILE VARIABLE,
(OUT,IN,INDICE,ARCH_DATOS,CONTROL_ARCHIVO)              FILE,
CADENA_LEIDA                                            CHAR(1000) VAR,
(NUMERO_ACTIVIDAD,NUMERO_ACTIVIDAD_1)                   CHAR(12) VAR,
(POS1,LON1,POS2,LON2)                                   BIN FIXED,
VECTOR_GUAR_NUM(30)                                      BIN FIXED,
VECTOR_GUAR_AP2(30)                                      DEC FIXED,
```

```
LK                                                              CHAR(1) VAR,
DTR                                                             DEC FIXED,
(API_INV1,API_INV2,AP2)                                         DEC FIXED,
NREF                                                            BIN FIXED,
(POS_ABS,PTR_PF_CM,LON_CM,PF_CR,UL_N,POS_RF,VTA(5))             BIN FIXED,
RFS_CM                                                          CHAR(120) VAR,
01 INF,
   02 APU_KAIZ                                                  BIN FIXED,
   02 NO_NIV                                                    BIN FIXED,
   02 NO_NODO                                                   BIN FIXED,
   02 APU_PSOX_VAC(0)                                           BIN FIXED,
01 NODO,
   02 CAB_NODO,
      03 NUM_H                                                  BIN FIXED,
      03 NIV                                                    BIN FIXED,
      03 NC                                                     BIN FIXED,
      03 CN                                                     BIN FIXED,
      03 VEC_INA_1                                              BIN FIXED,
      03 VEC_INA_0                                              BIN FIXED,
   02 ENTRADA                                                   CHAR(1009) VAR,
01 BLQ_MOVI_PF,
   02 CADENA_LEIDA2                                             CHAR(1000) VAR,
01 BLQ_FACT,
   02 CADENA_FACT                                               CHAR(1000) VAR,
01 CONTROL_ARCH(1A),
   02 TIPO                                                      BIN FIXED,
   02 N_ARC                                                     BIN FIXED,
   02 LON_REG                                                   BIN FIXED,
   02 BLK_VAC(0)                                                BIN FIXED,
   02 MAX_DTR                                                   BIN FIXED,
   02 NOMBRE_ARC                                                CHAR(20) VAR,
GOTOXY ENTRY(BIN FIXED,BIN FIXED),
PTOR_VAC(0) ENTRY(CHAR(20) VAR,DEC FIXED),
TRANSFERIA ENTRY(FILE VARIABLE,CHAR(120) VAR,DEC FIXED,BIN FIXED,DEC FIXED),
BUSCA ENTRY(FILE VARIABLE,CHAR(120) VAR,BIT(1),BIN FIXED,CHAR(120) VAR,
BIN FIXED,BIN FIXED,1,2,3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,
3 BIN FIXED,3 BIN FIXED,3 BIN FIXED,2 CHAR(1009) VAR,BIN FIXED,
(5) BIN FIXED,BIN FIXED,BIN FIXED,1,2 BIN FIXED,2 BIN FIXED,
2 BIN FIXED,2 BIN FIXED);

TESAURO = INDICE;
NUMERO_INDICE = 13;
NUMERO_ARC = 1;
IF VERIFY(NUMERO_ACTIVIDAD,"1234567890") = 0 THEN DO;
   DO I = 1 TO 9 - LENGTH(NUMERO_ACTIVIDAD);
      NUMERO_ACTIVIDAD = " " || NUMERO_ACTIVIDAD;
   END;
END;

READ FILE(CONTROL_ARCHIVO) INTO(CONTROL_ARCH) KEY(1);

CALL ABRE_INDICE;
CALL ABRE_ARCHIVO;
CALL BUSCA(TESAURO,CLAVE,RA_FNC,POS_ABS,RFS_CM,PF_CM,LON_CM,NODO,
PF_CR,VTA,BLK_N,POS_RF,INF);
CONT_AU = RANK(SUBSTR(ENTRADA,POS_ABS+5,1))*256 +
RANK(SUBSTR(ENTRADA,POS_ABS+6,1));
NUMERO_ACTIVIDAD_1 = "FA"||NUMERO_ACTIVIDAD;
POS1 = 1;
LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
POS1 = POS1 + LON1;
LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
POS1 = POS1 + LON1;
```

```
LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
POS1 = POS1 + LON1;
LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
MONTO_BS = DECIMAL(SUBSTR(CADENA_LEIDA,POS1+1,LON1-1),10,2);
POS1 = POS1 + LON1;
LON1 = RANK(SUBSTR(CADENA_LEIDA,POS1,1));
FLETE_BS = DECIMAL(SUBSTR(CADENA_LEIDA,POS1+1,LON1-1),10,2);
MONT_PAR = MONTO_BS / DECIMAL(CONT_AU,10,2);
FLETE_PAR = FLETE_BS / DECIMAL(CONT_AU,10,2);
J = 0;
CONTINUE;
    J = J + 1;
    CLAVE2 = CLAVE || CHAR(J);
CALL BUSCA(TESAURO,CLAVE2,RA_ENC,POS_ARS,RES_CM,PF_CM,LON_CM,MODO,
    PF_CB,UTA,IDENT_N,POS_RF,INF);
    NUM_AUT = RANK(SUBSTR(ENTRADA,POS_ARS+3,1))*256 +
    RANK(SUBSTR(ENTRADA,POS_ARS+4,1));
    AP2 = DEC(RANK(SUBSTR(ENTRADA,POS_ARS+7,1)))*10000 +
    RANK(SUBSTR(ENTRADA,POS_ARS+8,1))*100 +
    RANK(SUBSTR(ENTRADA,POS_ARS+9,1));
    VECTOR_CHAR_NUM(J) = NUM_AUT;
    VECTOR_CHAR_AP2(J) = AP2;
    IF J < CONT_AU THEN
       GOTO CONTINUE;
NUMERO_INDICE = 11;
CALL ABRE_INDICE;
DO I = 1 TO CONT_AU;
    CLAVE2 = CHAR(VECTOR_CHAR_NUM(I));
CALL BUSCA(TESAURO,CLAVE2,RA_ENC,POS_ARS,RES_CM,PF_CM,LON_CM,MODO,
    PF_CB,UTA,IDENT_N,POS_RF,INF);
    DIR = DEC(RANK(SUBSTR(ENTRADA,POS_ARS+2,1)))*10000 +
    RANK(SUBSTR(ENTRADA,POS_ARS+3,1))*100 +
    RANK(SUBSTR(ENTRADA,POS_ARS+4,1));
    READ FILE(ARCH_DATOS) INTO(BLQ_MONO_PP) KEY(DIR);
    POS = 1;
    LON = RANK(SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,POS,1));
    DO WHILE(LON ^= 255);
       POS = POS + LON;
       LON = RANK(SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,POS,1));
    END;
    CADENA_LEIDA2 = SUBSTR(BLQ_MONO_PP.CADENA_LEIDA2,1,POS-1) ||
    ASCII(LENGTH(NUMERO_ACTIVIDAD)+1) || NUMERO_ACTIVIDAD || ||
    ASCII(LENGTH(CHAR(MONT_PAR))+1) || CHAR(MONT_PAR) ||
    ASCII(LENGTH(CHAR(FLETE_PAR))+1) || CHAR(FLETE_PAR) ||
    SUBSTR(CADENA_LEIDA2,POS);
    WRITE FILE(ARCH_DATOS) FROM(BLQ_MONO_PP) KEYFROM(DIR);
       READ FILE(ARCH_DATOS) INTO(BLQ_MONO_PP) KEY(DIR);
       POS1 = 1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
  DO WHILE(NUMERO_ACTIVIDAD || ^= SUBSTR(CADENA_LEIDA2,POS1+1,LON1-1));
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
  END;
       POS1 = POS1 + LON1;
       LON1 = RANK(SUBSTR(CADENA_LEIDA2,POS1,1));
       POS2 = POS1 + LON1;
       LON2 = RANK(SUBSTR(CADENA_LEIDA2,POS2,1));
END;
NUMERO_INDICE = 14;
NUMERO_ARC = 4;
CALL ABRE_ARCHIVO;
CALL ABRE_INDICE;
CADENA_PAC1 = CADENA_LEIDA || ASCII(255);
```

```
FACTURA:  - NUMERO_ACTIVIDAD;
NOMBRE_ARCHIVO = CONTROL_ARCH(I).NOMBRE_ARC;
CALL PUSH_VACIO(NOMBRE_ARCHIVO,API_INV1);
WRITE FILE(ARCH_DATOS) FROM(BLQ_FACT) KEYFROM(API_INV1);
CLAVE2 = NUMERO_ACTIVIDAD;
API_INV2 = 1;
NREF = CONT_AU;
CALL INSERTA(TESAURO,CLAVE2,API_INV1,NREF,API_INV2);
DO I = 1 TO CONT_AU;
   CLAVE = CLAVE2 || CHAR(I);
   NREF = VECTOR_GUAR_NUM(I);
   API_INV2 = VECTOR_GUAR_API(I);
   CALL INSERTA(TESAURO,CLAVE,API_INV1,NREF,API_INV2);
END;

ABRE_INDICE: PROCEDURE;

   CLOSE FILE(INDICE);
   I = 1;
   DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_INDICE);
      I = I + 1;
   END;
   OPEN FILE(INDICE) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
   DIRECT KEYED ENV(REGSIZE(1024));

END; /*  ABRE_INDICE  */

ABRE_ARCHIVO: PROCEDURE;

   I = 1;
   DO WHILE(CONTROL_ARCH(I).N_ARC ^= NUMERO_ARC);
      I = I + 1;
   END;
   LONGITUD = CONTROL_ARCH(I).LON_REG;
   CLOSE FILE(ARCH_DATOS);
   OPEN FILE(ARCH_DATOS) TITLE(CONTROL_ARCH(I).NOMBRE_ARC) UPDATE
   DIRECT KEYED ENV(REGSIZE(LONGITUD));

END;    /*  ABRE_ARCHIVO    */

END;    /*   FACTURA_AUT     */
```