Implementación de Tipos de Datos Abstractos con memoria dinámica

Para poder almacenar los TDA con memoria dinámica se necesita crear un nodo de tipo registro el cual va a contener una parte de **INFO** y otra **SIG**

INFO SIG

INFO: donde se almacenarán los datos propios a guardar
SIG: de tipo puntero con la posición al siguiente elemento en la estructura

Declaración

```
TYPE
T_DATO= RECORD

        NOM: STRING[40];

        TEL: STRING[15];

        END;

T_PUNT=^T_NODO;

T_NODO=RECORD

        INFO:T_DATO;

        SIG:T_PUNT;

        end;
```

Implementación de pilas con memoria dinámica

```
unit pilas;
interface
 TYPE
T_DATO= RECORD
            NOM: STRING[40];
            TEL: STRING[15];
            END;
T_PUNT=^T_NODO;
T_NODO=RECORD
            INFO:T_DATO;
            SIG:T_PUNT;
            end;
T_PILA= RECORD
            TOPE: T_PUNT;
            TAM: WORD;
            END;
PROCEDURE CREARPILA(VAR P:T_PILA);
PROCEDURE  APILAR (VAR P:T_PILA; X:T_DATO);
FUNCTION PILA_LLENA (VAR P:T_PILA): BOOLEAN;
FUNCTION PILA_VACIA (VAR P:T_PILA): BOOLEAN;
PROCEDURE DESAPILAR (VAR P:T_PILA; VAR X:T_DATO);
implementation
PROCEDURE CREARPILA(VAR P:T_PILA);
BEGIN
  P.TAM:=0;
  P.TOPE:=NIL;
END;
PROCEDURE APILAR (VAR P:T_PILA; X:T_DATO);
VAR DIR:T_PUNT;
BEGIN
NEW (DIR);
DIR^.INFO:= X;
```

```
DIR^.SIG:= P.TOPE;

P. TOPE:=DIR;

INC(P.TAM)

END;


FUNCTION PILA_LLENA (VAR P:T_PILA): BOOLEAN;

BEGIN

PILA_LLENA:= GETHEAPSTATUS.TotalFree < SIZEOF(T_NODO) ;

END;

FUNCTION PILA_VACIA (VAR P:T_PILA): BOOLEAN;

BEGIN

PILA_VACIA:= P.TAM=0;

END;

PROCEDURE DESAPILAR (VAR P:T_PILA;VAR X:T_DATO);

VAR dir:t_punt;

BEGIN

  X:=  P.TOPE^.INFO;

  dir:=P.TOPE;

  P.TOPE:=P.TOPE^.sig;

  DISPOSE (dir);

  DEC(P.TAM)

 END;

END.
```

Implementación de colas con memoria dinámica

```
unit COLAS;

interface

TYPE

T_DATO= RECORD

        NOM: STRING[40];

        TEL: STRING[15];

        END;

T_PUNT=^T_NODO;

T_NODO=RECORD

        INFO:T_DATO;

        SIG:T_PUNT;

        end;

T_COLA= RECORD

        FRENTE,FINAL: T_PUNT;

        TAM: WORD;

        END;

PROCEDURE CREARCOLA(VAR Q:T_COLA);

PROCEDURE  ENCOLAR (VAR Q:T_COLA; X:T_DATO);

FUNCTION COLA_LLENA (VAR Q:T_COLA): BOOLEAN;

FUNCTION COLA_VACIA (VAR Q:T_COLA): BOOLEAN;

PROCEDURE DESENCOLAR (VAR Q:T_COLA; VAR X:T_DATO);

implementation

PROCEDURE CREARCOLA(VAR Q:T_COLA);

BEGIN

        Q.TAM:=0;

        Q.FRENTE:=NIL;

        Q.FINAL:= NIL;

END;
```

```
PROCEDURE ENCOLAR (VAR Q:T_COLA; X:T_DATO);

VAR DIR:T_PUNT;

BEGIN

NEW (DIR);

DIR^.INFO:= X;

DIR^.SIG:= NIL;

IF Q.FRENTE = NIL THEN  Q.FRENTE:=DIR

                        ELSE Q.FINAL^.SIG:= DIR;

Q.FINAL:= DIR; INC(Q.TAM);

END;

FUNCTION COLA_LLENA (VAR Q:T_COLA): BOOLEAN;

BEGIN

COLA_LLENA:= GETHEAPSTATUS.TotalFree < SIZEOF(T_NODO) ;

END;

FUNCTION COLA_VACIA (VAR Q:T_COLA): BOOLEAN;

BEGIN

COLA_VACIA:= Q.TAM =0;

END;

PROCEDURE DESENCOLAR (VAR Q:T_COLA;VAR X:T_DATO);

VAR act: t_punt;

BEGIN

X:=  Q.FRENTE^.INFO;

ACT:=Q.FRENTE;

Q.FRENTE:=ACT^.SIG;

DISPOSE (ACT);

IF Q.FRENTE = NIL THEN Q.FINAL:= NIL;

DEC(Q.TAM)

END;

END.
```

Implementación de listas secuenciales ordenadas por contenido con memoria dinámica

```
UNIT U_LISTA_SEC_MEM_DIN;

 INTERFACE
 CONST
   N=10;
   TYPE

    T_DATO =  RECORD
              NOM:STRING[40];
              EDAD: BYTE;
              CANT_MAT: BYTE;
             end;


      T_PUNT = ^T_NODO;

      T_NODO = RECORD
                INFO:T_DATO;
                SIG:T_PUNT;
               END;


      T_LISTA  = RECORD
                   CAB,ACT: T_PUNT;
                   TAM: 0..N;
                  END;


  PROCEDURE CREARLISTA(VAR L:T_LISTA);
  PROCEDURE AGREGAR (VAR L:T_LISTA; X:T_DATO);
  FUNCTION TAMANIO (VAR L:T_LISTA): BYTE;
  FUNCTION LISTA_LLENA (VAR L:T_LISTA): BOOLEAN;
  FUNCTION LISTA_VACIA (VAR L:T_LISTA): BOOLEAN;
  PROCEDURE ELIMINARLISTA (VAR L:T_LISTA;BUSCADO: STRING; VAR X:T_DATO);
  PROCEDURE SIGUIENTE(VAR L:T_LISTA);
  PROCEDURE PRIMERO (VAR L:T_LISTA);
  FUNCTION FIN  (L:T_LISTA): BOOLEAN;
  PROCEDURE RECUPERAR (L:T_LISTA;  VAR X:T_DATO);


implementation

PROCEDURE CREARLISTA(VAR L:T_LISTA);
BEGIN
             L.TAM:=0;
      L.CAB:=NIL;
END;

PROCEDURE AGREGAR (VAR L:T_LISTA; X:T_DATO);
VAR DIR, ANT: T_PUNT;
BEGIN
NEW (DIR);
DIR^.INFO:= X;
IF (L.CAB= NIL) OR (L.CAB^.INFO.NOM > X.NOM) THEN
                          BEGIN
                         DIR^.SIG:= L.CAB;
                          L. CAB:=DIR;
                            END
     ELSE
      BEGIN
       ANT:= L.CAB;
       L.ACT:= L.CAB^.SIG;
      WHILE (L.ACT <> NIL) AND (L.ACT^.INFO.NOM < X.NOM) DO
                BEGIN
                ANT:= L.ACT;
                L.ACT:= L.ACT^.SIG
               END;
           DIR^.SIG:= L.ACT;
           ANT^.SIG:= DIR;
      END;
INC(L.TAM)
END;
```

```
PROCEDURE SIGUIENTE(VAR L:T_LISTA);
BEGIN
L.ACT:= L.ACT^.SIG;
END;

PROCEDURE PRIMERO (VAR L:T_LISTA);
BEGIN
L.ACT:= L.CAB;
END;

FUNCTION FIN  (L:T_LISTA): BOOLEAN;
BEGIN
FIN:=L.ACT = NIL;
END;

FUNCTION TAMANIO (VAR L:T_LISTA): BYTE;
BEGIN
  TAMANIO:= L.TAM;
end;

FUNCTION LISTA_LLENA (VAR L:T_LISTA): BOOLEAN;
BEGIN
LISTA_LLENA:= GETHEAPSTATUS.TotalFree < SIZEOF(T_DATO) ;
END;

FUNCTION LISTA_VACIA (VAR L:T_LISTA): BOOLEAN;
BEGIN
LISTA_VACIA:= L.TAM=0;
END;

PROCEDURE RECUPERAR (L:T_LISTA;  VAR X:T_DATO);
BEGIN
X:= L.ACT^.INFO;
END;

//este procedimiento eliminar se invoca luego de haber chequeado que el elemento a eliminar existe

PROCEDURE ELIMINARLISTA (VAR L:T_LISTA;BUSCADO: STRING; VAR X:T_DATO);
VAR   ANT: T_PUNT;
BEGIN
     IF (L.CAB^.INFO.NOM= BUSCADO) THEN
                          BEGIN
                           X:=  L.CAB^.INFO;
                           L.ACT:=L.CAB;
                           L.CAB:=L.CAB^.SIG;
                           DISPOSE (L.ACT);
                          END
     ELSE
      BEGIN
       ANT:=L.CAB;
       L.ACT:= L.CAB^.SIG;
       WHILE (L.ACT <> NIL) AND (L.ACT^.INFO.NOM< X.NOM) DO
                  BEGIN
                          ANT:=L.ACT;
                   L.ACT:= L.ACT^.SIG;
                  END;
          X:= L.ACT^.INFO;
          ANT^.SIG:= L.ACT^.SIG;
          DISPOSE (L.ACT);
       END ;
      DEC(L.TAM);
END;

  END.
```

Otros subprogramas que se necesitarán para buscar y recorrer una lista

```
PROCEDURE MUESTRA_LISTA(L:T_LISTA);     //LISTADO
  VAR
    X:T_DATO;
BEGIN
PRIMERO(L);
WHILE NOT FIN(L) DO
  BEGIN
   RECUPERAR(L,X);
  GOTOXY(1,POSICION+1); WRITE (X.NOM);
   GOTOXY(30,POSICION+1); WRITE (X.EDAD);
   GOTOXY(50,POSICION+1); WRITE (X.CANT_MAT);
   SIGUIENTE(L);
  END;


PROCEDURE BUSCAR (VAR L:T_LISTA; BUSCADO:STRING; VAR ENC:BOOLEAN);
 VAR X:T_DATO;
BEGIN
ENC:=FALSE;
PRIMERO (L);
WHILE not FIN (L) AND (NOT ENC) DO
BEGIN
  RECUPERAR (L, X);
  IF X.NOM = BUSCADO THEN ENC:=TRUE
                ELSE  SIGUIENTE (L)
END;
END;
```

NOTA: el tipo de dato del buscado dependerá de lo que se está buscando