**program Ejercicio1;**

type

lista = ^nodo;

nodo = record

dato:integer;

ant:lista;

sig:lista;

end;

LD = record

pri:lista;

ult:lista;

end;

procedure agregarAtras(id:integer; var l:LD);

var

nuevo:lista;

begin

new(nuevo);

nuevo^.dato:=id;

nuevo^.ant:=nil;

nuevo^.sig:=nil;

if (l.pri=nil) then

begin

l.pri:=nuevo;

l.ult:=nuevo;

end

else

begin

l.ult^.sig:=nuevo;

nuevo^.ant:=l.ult;

l.ult:=nuevo;

end;

end;

procedure agregarAdelanteMedioAtras(id:integer; var l:LD);

var

ant, act, nuevo: lista;

begin

act:= l.pri;

new(nuevo);

nuevo^.dato:=id;

nuevo^.sig:=nil;

nuevo^.ant:=nil;

while (act <> nil) and (id > act^.dato) do

begin

ant:= act;

act:= act^.sig;

end;

if (act = l.pri) then begin {agrega primero}

nuevo^.sig:= l.pri;

l.pri := nuevo;

end

else begin {es en el medio o al final}

ant^.sig := nuevo;

nuevo^.sig := act;

end;

end;

procedure cargar(var L:LD);

var

id:integer;

begin

l.pri:=nil;

l.ult:=nil;

readln(id);

while (id <> 999) do

begin

//agregarAtras(id,L); o agregarAdelanteMedioAtras(id,L);

// depende si te lo dan ordenado o no.

readln(id);

end;

end;

procedure recorrerAscendente(l:LD);

begin

while (l.pri<>nil) do

begin

writeln(l.pri^.dato);

l.pri:=l.pri^.sig;

end;

end;

procedure recorrerDescendente(l:LD);

begin

while (l.ult<>nil) do

begin

writeln(l.ult^.dato);

l.ult:=l.ult^.ant;

end;

end;

//Programa principal

var

l:LD;

begin

cargar(L);

recorrerAscendente(L);

recorrerDescendente(L);

readln();

readln();

end.

**program Ejercicio2;**

TYPE

cadena = string[30];

alumnos = ^nodo;

alumno = record

numero : integer;

dni : integer;

apellido: cadena;

nombre : cadena;

end;

nodo = record

dato: alumno;

sig\_num: alumnos;

sig\_ape: alumnos;

sig\_dni: alumnos;

end;

lista\_doble = record

pri\_ape: alumnos;

pri\_num: alumnos;

pri\_dni: alumnos;

end;

{Proceso que lee la información de un alumno}

procedure leer\_alumno (var a: alumno);

begin

writeln;

writeln('---- NUEVO ALUMNO ----');

writeln;

write('DNI: ');

readln(a.dni);

if (a.dni <> 0) then begin

write('Numero: ');

readln(a.numero);

write('Apellido: ');

readln(a.apellido);

write('Nombre: ');

readln(a.nombre);

end;

end;

{Proceso que inserta un alumno a la lista respetando el orden por apellido}

procedure agregar\_x\_apellido (var l: lista\_doble; nue: alumnos);

var

ant, act: alumnos;

begin

act:= l.pri\_ape;

while (act <> nil) and (nue^.dato.apellido>act^.dato.apellido) do begin

ant:= act;

act:= act^.sig\_ape;

end;

if (act = l.pri\_ape) then begin {agrega primero}

nue^.sig\_ape:= l.pri\_ape;

l.pri\_ape := nue;

end

else begin {es en el medio o al final}

ant^.sig\_ape := nue;

nue^.sig\_ape := act;

end;

end;

{Proceso que inserta un alumno a la lista respetando el orden por número de alumno}

procedure agregar\_x\_numero (var l: lista\_doble; nue: alumnos);

var

ant, act: alumnos;

begin

act:= l.pri\_num;

while (act <> nil) and (nue^.dato.numero > act^.dato.numero) do begin

ant:= act;

act:= act^.sig\_num;

end;

if (act = l.pri\_num) then begin {agrega primero}

nue^.sig\_num:= l.pri\_num;

l.pri\_num := nue;

end

else begin {es en el medio o al final}

ant^.sig\_num := nue;

nue^.sig\_num := act;

end;

end;

{Proceso que inserta un alumno a la lista respetando el orden por dni}

procedure agregar\_x\_dni (var l: lista\_doble; nue: alumnos);

var

ant, act: alumnos;

begin

act:= l.pri\_dni;

while (act <> nil) and (nue^.dato.dni > act^.dato.dni) do begin

ant:= act;

act:= act^.sig\_dni;

end;

if (act = l.pri\_dni) then begin {agrega primero}

nue^.sig\_dni:= l.pri\_dni;

l.pri\_dni := nue;

end

else begin {es en el medio o al final}

ant^.sig\_dni := nue;

nue^.sig\_dni := act;

end;

end;

{Proceso que agrega un alumno a la lista por ambos órdenes}

procedure agregar (var l: lista\_doble; a:alumno);

var

nue: alumnos;

begin

{Se crea el nodo una vez}

new(nue);

nue^.dato := a;

nue^.sig\_ape := nil;

nue^.sig\_num := nil;

nue^.sig\_dni := nil;

{Agrego por los dos órdenes}

agregar\_x\_apellido(l, nue);

agregar\_x\_numero (l, nue);

agregar\_x\_dni (l, nue);

end;

{Proceso que va leyendo la información de los alumnos y los agrega a la lista}

procedure crearlista (var l: lista\_doble);

var

alu: alumno;

Begin

{Inicializo la lista}

l.pri\_num:= nil;

l.pri\_ape:= nil;

l.pri\_dni:= nil;

leer\_alumno(alu);

while (alu.dni <> 0) do begin

agregar(l, alu);

leer\_alumno(alu);

end;

end;

{VARIABLES DEL PROGRAMA PRINCIPAL}

var

l : lista\_doble;

Begin {PROGRAMA PRINCIPAL}

crearlista(l);

{while (l.pri\_dni <> nil) do begin

writeln('---------------------------------');

writeln(l.pri\_dni^.dato.numero);

writeln(l.pri\_dni^.dato.dni);

writeln(l.pri\_dni^.dato.apellido);

writeln(l.pri\_dni^.dato.nombre);

l.pri\_dni:= l.pri\_dni^.sig\_dni;

end;

readln();

readln();} // -> para probar si anda.

end.

**program Ejercicio3;**

type

lista= ^nodo;

nodo = record

id:integer;

jugador:string[20];

sig:lista;

ant:lista;

end;

procedure cargar(var l:lista);

var

jug:nodo;

ult:lista;

begin

leer(jug^.jugador);

while (jug^.jugador <> 999) do

begin

agregar(jug,l,ult);

leer(jug^.jugador);

end;

end;

procedure jugar(var l:lista);

var

begin

while (l.pri^.id <> tirarDados )do