**function** menorEnArbol(arbol: puntero): integer;

begin

if(arbol <> nil) then begin

if(arbol^.menor <> nil) then

menorEnArbol := menorEnArbol(arbol^.menor)

else

menorEnArbol := arbol^.valor;

end else

menorEnArbol := 0;

end;

**function** BuscaAnterior(lista: p\_nodo; valor: integer): p\_nodo;

begin

if (lista <> nil) and (lista^.valor = valor) then

BuscaAnterior := lista

else if (lista^.next <> nil) then begin

if (lista^.next^.valor <> valor) then

BuscaAnterior := BuscaAnterior(lista^.next, valor)

else

BuscaAnterior := lista;

end else

BuscaAnterior := nil;

end;

**Procedure** InsertarEnArbol(var Arbol:TipoArbol; Nodo:TipoArbol);

begin

if (Arbol = Nil) then

Arbol := Nodo

else

begin

if (Nodo^.valor < Arbol^.valor) then

InsertarEnArbol(Arbol^.menor,Nodo)

else

InsertarEnArbol(Arbol^.mayor,Nodo);

end;

end;

**Procedure** CargarArbol(var Arbol:TipoArbol);

var

i:integer;

Nodo:TipoArbol;

begin

i:= 0; //PARA ENTRAR EN EL BUCLE

while (i <> -1) do

begin

CrearNodo(Nodo);

InsertarEnArbol(Arbol,Nodo);

writeln('ingrese un valor distinto a -1 para seguir agregando nodos');

readln(i);

end;

end;

**function** longitud(nodo: TipoArbol): integer;

var longMayores, longMenores, raiz: integer;

begin

if(nodo <> nil) then begin

longMayores := longitud(nodo^.mayor);

longMenores := longitud(nodo^.menor);

raiz:= 1;

longitud:= longMayores + longMenores + raiz;

end else

longitud := 0;

end;

**procedure** ImprimirArbolAscendente(arbol:PuntArbol);

begin

if (arbol <> nil) then

begin

ImprimirArbolAscendente(arbol^.menor);

writeln('nro\_facturas: ',arbol^.nro\_factura);

writeln('facturas\_impagas: ',arbol^.facturas\_impagas);

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

ImprimirArbolAscendente(arbol^.mayor);

end;

end;

**procedure** imprimirDescendente (arbol:PuntArbol);

begin

if (arbol <> nil) then

begin

writeln (arbol^.valor);

imprimirAscendente(arbol^.mayores);

imprimirAscendente(arbol^.menores);

end;

end;

**procedure** imprimirPostOrder (arbol:PuntArbol);

//no se imprime un padre si no se han impreso todos sus hijos

begin

if (arbol <> nil) then

begin

imprimirAscendente(arbol^.mayores);

imprimirAscendente(arbol^.menores);

writeln (arbol^.valor); //padre

end;

end;

**procedure** imprimirPreOrder (arbol:PuntArbol);

//no se imprime un nodo si no se ha impreso su padre

begin

if (arbol <> nil) then

begin

writeln (arbol^.valor);

imprimirAscendente(arbol^.menores);

imprimirAscendente(arbol^.mayores);

end;

end;

**function** SumaArbol (nodo: PuntArbol): integer; //suma todos los valores de cada nodo

begin

if (nodo <> nil) then

SumaArbol:= nodo^.'VALOR QUE TENGA EL TYPE' + SumaArbol (nodo^.menor) + SumaArbol (nodo^.mayor) //sumas mientras sea dist a nil

else

SumaArbol:= 0; //si es nil, suma 0

end;

**Procedure** CrearNodo(var Nodo:TipoArbol);

begin

New(Nodo);

writeln('inserte valor al nodo:');

readln(Nodo^.valor);

Nodo^.mayor := nil;

Nodo^.menor := nil;

end;

**Procedure** ImprimirListaEnOrden(cursor: p\_nodo);

begin

if cursor <> nil then begin

writeln(cursor^.valor);

ImprimirListaEnOrden(cursor^.sig);

end;

end;

**Procedure** ImprimirListaAlReves(cursor: p\_nodo);

begin

if cursor <> nil then begin

ImprimirListaAlReves(cursor^.sig);

writeln(cursor^.valor);

end;

end;

**Procedure** CrearListaCircular(var lista: puntero);

var input: integer;

nuevoNodo: puntero;

begin

writeln('Meteme un valorcito');

readln(input);

if(input <> -1) then begin

CrearNodo(nuevoNodo, input);

AgregarAlFinal(lista, lista, nuevoNodo);

CrearListaCircular(lista);

end else

writeln('Fin de la carga');

end;

**Procedure** AgregarAlFinal(var lista: puntero; cursor, nodo: puntero);

begin

if(lista = nil) then begin

lista := nodo;

nodo^.sig := lista;

end else if(cursor^.sig = lista) then begin

// Llegar al inicio de nuevo

cursor^.sig := nodo;

nodo^.sig := lista;

end else

AgregarAlFinal(lista, cursor^.sig, nodo);

end;

**Procedure** ImprimirListaEnOrden(lista, cursor: puntero);

begin

if(lista <> nil) then begin

writeln(cursor^.valor);

if cursor^.sig <> lista then

ImprimirListaEnOrden(lista, cursor^.sig);

end else

writeln('La lista circular está vacía...');

end;

[21:15, 25/10/2021] Laza Ventancor: Imprimir lista circular con recursión

[21:15, 25/10/2021] Laza Ventancor: Cursor comienza en lista

**Procedure** Desvincular(var lista, nodo: puntero);

begin

// Si la lista <> nil

if(lista = nodo) then begin

lista := lista^.sig;

// Hay un solo nodo en la lista

if(lista <> nil) then

lista^.anterior := nil;

end else if (nodo^.siguiente = nil) then

// El nodo esta al final

nodo^.anterior^.siguiente := nil

else begin

// El nodo esta al medio

nodo^.anterior^.siguiente := nodo^.siguiente;

nodo^.siguiente^.anterior := nodo^.anterior;

end;

nodo^.siguiente := nil;

nodo^.anterior := nil;

end;

[21:35, 25/10/2021] Laza Ventancor: Desvincular de lista doblemente vinculada

**function** BuscaAnterior(lista: p\_nodo; valor: integer): p\_nodo;

begin

if (lista = nil) or (lista^.valor = valor) then

BuscaAnterior := lista

else if (lista^.siguiente <> nil) then begin

if (lista^.siguiente^.valor <> valor) then

BuscaAnterior := BuscaAnterior(lista^.siguiente, valor)

else

BuscaAnterior := lista;

end else

BuscaAnterior := nil;

end;

[21:48, 25/10/2021] Laza Ventancor: Busca anteorior en lista simple o doble...