**LISTA DE LISTAS CLASE 1**

**Type de lista de listas**

ListaInt = ^nodoInt;

nodoInt = record

mensajetw:string;

sig:ListaInt;

end;

DatoListaOrd= record

list:ListaInt;

nombre:string;

end;

ListaOrd = ^nodo;

nodo = record

datos:DatoListaOrd;

sig:ListaOrd;

end;

// Arbol de enteros

arbol= ^nodoA;

nodoA = Record

dato: integer;

HI: arbol;

HD: arbol;

End;

**agregar adelante**

{agregarAdelante - Agrega nro adelante de l}

Procedure agregarAdelante(var l: listaTweets; t: tweet);

var

aux: listaTweets;

begin

new(aux);

aux^.dato := t;

aux^.sig := l;

l:= aux;

end;

**IMPRIMIR LISTA**

{imprimir - Muestra en pantalla el tweet}

procedure imprimir(t: tweet);

begin

with (t) do begin

write('Tweet del usuario @', nombreUsuario, ' con codigo ',codigoUsuario, ': ', mensaje, ' RT:');

if(esRetweet)then

writeln(' Si')

else

writeln('No ');

end;

end;

{imprimirLista - Muestra en pantalla la lista l}

procedure imprimirLista(l: listaTweets);

begin

while (l <> nil) do begin

imprimir(l^.dato);

l:= l^.sig;

end;

end;

**AGREGAR ORDENADO**

{agregarElemento - Resuelve la inserción de la estructura ordenada}

procedure agregarOrdenado(var pri:listaTweets; t:tweet);

var

nuevo, anterior, actual: listaTweets;

begin

new (nuevo);

nuevo^.dato:= t;

nuevo^.sig := nil;

if (pri = nil) then

pri := nuevo

else

begin

actual := pri;

anterior := pri;

while (actual<>nil) and (actual^.dato.nombreUsuario < nuevo^.dato .nombreUsuario) do begin

anterior := actual;

actual:= actual^.sig;

end;

if (anterior = actual) then

pri := nuevo

else

anterior^.sig := nuevo;

nuevo^.sig := actual;

end;

end;

**AGREGAR AL FINAL**

procedure AgregarAlFinal (var pri: ListaInt; p: string);

var act, nue : ListaInt;

begin

new (nue);

p:=nue^.mensajetw;

nue^.sig := NIL;

if pri <> Nil then begin

act := pri ;

while (act^.sig <> NIL ) do act := act^.sig ;

act^.sig := nue ;

end

else

pri:= nue;

end;

CLASE 2 RECURSION

FACTORIAL

function factorial(num:integer):real;

begin

if(num<=1)then begin

factorial:=1

end

else

factorial:=num\*factorial(num-1);

end;

POTENCIA

function potencia1(x,n:integer):real;

begin

if(n=0)then

potencia1:=1

else

potencia1:=x\*potencia1(x,n-1);

end;

DIGITO MAXIMO

function digitoMaximo(n: integer):integer;

var

dig: integer; max:integer;

begin

if(n=0) then begin

max:=0;

end

else begin

max:=digitoMaximo(n div 10);

dig:= n mod 10;

if (dig > max) then

max:= dig;

end;

digitoMaximo:=max;

end;

BUSQUEDA DICOTOMICA

procedure BusquedaDicotomica(v:vector; num:integer; dimL:dim; var pri:dim; var ult:dim; var pos:dim);

var

medio:dim;

begin

if (v[ult]<num)then

writeln('No se encontró el elemento ')

else begin

medio:= (pri + ult) div 2;

if num = v[medio] then

pos:=medio

else begin

if (num < v[medio]) then begin

ult:=medio-1;

BusquedaDicotomica(v,num,dimL,pri,ult,pos);

end

else begin

pri:=medio+1;

BusquedaDicotomica(v,num,dimL,pri,ult,pos);

end;

end;

end;

end;

**MAXIMO DENTRO DE UN VECTOR**

Function maximo (v:vector; dimL: dim; max:integer; i:integer): integer;

Begin

if (i>=dimL) then begin //CONDICION BASE- TERMINA SU FUNCION

maximo := max;

end

else begin

if (v[i] > max) then begin

max:= v[i];

end;

max:= maximo (v,dimL,max,i+1);

end;

maximo:= max; //VAYA GUARDANDO CUAL ES EL MAXIMO, ya que como es función no se puede pasar por referencia.

end;

**SUMA RECURSIVA DE VALORES DENTRO DE UN VECTOR**

function sumavector (v:vector; dimL:dim; suma:integer; i:integer): integer;

begin

if (i>=dimL) then begin

sumavector:= suma; //condicion base, aca terminaria

end

else begin

suma:= suma + v[i];

suma:= sumavector(v,dimL,suma,i+1);

end;

sumavector:= suma; //va guardando cual es la suma

end;

**MINIMO VALOR DE LISTA**

function minimo(L:Lista; min:integer):integer;

begin

if (L = nil) then

minimo:=min //caso base

else begin

if (L^.datos < min) then

min:= L^.datos;

min:= minimo(L^.sig,min);

end;

minimo:=min;

end;

**IMPRIMIR LISTA RECURSIVA**

procedure ImprimirRecursivo(L:Lista);

begin

if (L <> nil) then begin

write(L^.datos,' ');

ImprimirRecursivo(L^.sig);

end;

end;

**CLASE 3 ARBOLES**

**TYPE ARBOL**

**Agregar Arbol - Agrega un entero a un arbol ABO, sin repetidos }**

**Procedure insertar(var a: arbol; num:integer);**

**begin**

**if (a = nil)then**

**begin**

**new (a);**

**a^.dato:= num;**

**a^.HI := nil;**

**a^.HD := nil;**

**end**

**else**

**if a^.dato> num then**

**insertar(a^.HI, num)**

**else**

**if a^.dato< num then**

**insertar(a^.HD, num)**

**end;**

**IMPRIMIR POR NIVEL**

**function ContarElementos (l: listaNivel): integer;**

**var c: integer;**

**begin**

**c:= 0;**

**While (l <> nil) do begin**

**c:= c+1;**

**l:= l^.sig;**

**End;**

**contarElementos := c;**

**end;**

**Procedure AgregarAtras (var l, ult: listaNivel; a:arbol);**

**var nue:listaNivel;**

**begin**

**new (nue);**

**nue^.info := a;**

**nue^.sig := nil;**

**if l= nil then l:= nue**

**else ult^.sig:= nue;**

**ult:= nue;**

**end;**

**Procedure imprimirpornivel(a: arbol);**

**var**

**l, aux, ult: listaNivel;**

**nivel, cant, i: integer;**

**begin**

**l:= nil;**

**if(a <> nil)then begin**

**nivel:= 0;**

**agregarAtras (l,ult,a);**

**while (l<> nil) do begin**

**nivel := nivel + 1;**

**cant:= contarElementos(l);**

**write ('Nivel ', nivel, ': ');**

**for i:= 1 to cant do begin**

**write (l^.info^.dato, ' - ');**

**if (l^.info^.HI <> nil) then agregarAtras (l,ult,l^.info^.HI);**

**if (l^.info^.HD <> nil) then agregarAtras (l,ult,l^.info^.HD);**

**aux:= l;**

**l:= l^.sig;**

**dispose (aux);**

**end;**

**writeln;**

**end;**

**end;**

**end;**

**PREORDEN**

**Procedure PreOrden( a: arbol );**

**begin**

**if ( a <> nil ) then begin**

**write (a^.dato, ' ');**

**preOrden(a^.HI);**

**preOrden(a^.HD);**

**end;**

**end;**

**ENORDEN**

**Procedure enOrden( a: arbol );**

**begin**

**if ( a <> nil ) then begin**

**enOrden (a^.HI);**

**write (a^.dato, ' ');**

**enOrden (a^.HD)**

**end;**

**end;**

**POSTORDEN**

**Procedure PostOrden( a: arbol );**

**begin**

**if ( a <> nil ) then begin**

**PostOrden(a^.HI);**

**PostOrden(a^.HD);**

**write (a^.dato, ' ');**

**end;**

**end;**

**BUSCAR EN UN ARBOL Y DEVUELVE UN PUNTERO AL DATO ENCONTRADO**

**function buscar(a:arbol;numero:integer;var encontro:boolean):arbol;**

**begin**

**if(a<>nil)then begin**

**if(a^.dato=numero)then begin**

**encontro:=true;**

**buscar:=a;**

**end**

**else begin**

**buscar(a^.HI,numero,encontro);**

**buscar(a^.HD,numero,encontro);**

**end;**

**end;**

**if(encontro=false)then**

**buscar:=nil;**

**end;**

**VERMIN (BUSCAR MINIMO ENTRE LOS DATOS DEL ARBOL)**

**function vermin(a:arbol):integer;**

**begin**

**if(a<>nil)then begin**

**if(a^.HI <>nil)then begin**

**vermin:=vermin(a^.HI);**

**end**

**else**

**vermin:=a^.dato;**

**end;**

**end;**

**VERMAX (BUSCAR MAXIMO ENTRE LOS DATOS DEL ARBOL)**

**function vermax(a:arbol):integer;**

**begin**

**if(a<>nil)then begin**

**if(a^.HD <>nil)then begin**

**vermax:=vermax(a^.HD);**

**end**

**else**

**vermax:=a^.dato;**

**end;**

**end;**

**CLASE 4 ARBOLES**

**BUSQUEDA ACOTADA**

**procedure Acotado(a:arbol;inf:integer;sup:integer);**

**begin**

**if(a<>nil)then**

**if(a^.dato.dni>inf)then**

**if(a^.dato.dni<sup)then begin**

**with a^.dato do begin**

**writeln('el nombre es : ',nombre);**

**writeln('el apellido es : ',apellido);**

**writeln('el dni es : ',dni);**

**end**

**end**

**else**

**acotado(a^.HI,inf,sup)**

**else**

**acotado(a^.HD,inf,sup);**

**end;**

**BUSQUEDA ACOTADA VARIOS ELEMENTOS**

**procedure acotado(a:arbol;inf:integer;sup:integer;var cant:integer);**

**begin**

**if(a<>nil)then**

**if(a^.info.dni>=inf)then**

**if(a^.info.dni<=sup)then begin**

**cant:=cant+1;**

**acotado(a^.HI,inf,sup,cant);**

**acotado(a^.HD,inf,sup,cant);**

**end**

**else acotado(a^.HI,inf,sup,cant)**

**else acotado(a^.HD,inf,sup,cant);**

**end;**

**borrar elemento**

**function vermin(a:arbol):integer;**

**begin**

**if(a<>nil)then begin**

**if(a^.HI <>nil)then begin**

**vermin:=vermin(a^.HI);**

**end**

**else**

**vermin:=a^.dato;**

**end;**

**end;**

**function buscar(a:arbol;numero:integer;var encontro:boolean):arbol;**

**begin**

**if(a<>nil)then begin**

**if(a^.dato=numero)then begin**

**encontro:=true;**

**buscar:=a;**

**end**

**else begin**

**if(numero<a^.dato)then**

**buscar(a^.HI,numero,encontro)**

**else**

**if(numero>a^.dato)then**

**buscar(a^.HD,numero,encontro);**

**end;**

**end;**

**if(encontro=false)then**

**buscar:=nil;**

**end;**

**procedure borrarelemento(var a:arbol;dato:integer;var resultado:boolean);**

**var**

**auxa:arbol; encontro:boolean;**

**begin**

**encontro:=false;**

**if(a=nil)then begin resultado:=false end**

**else if(a^.dato >dato)then begin borrarelemento(a^.HI,dato,resultado) end**

**else if(a^.dato <dato)then begin borrarelemento(a^.HD,dato,resultado) end**

**//encontramos dato**

**else if ((a^.HD = nil) and (a^.HI = nil) ) then begin**

**dispose(a);**

**a:=nil;**

**resultado:=true;**

**end**

**//tiene hijo der**

**else if ((a^.HD <> nil) and (a^.HI = nil) ) then begin**

**auxa:=a;**

**a:=a^.HD;**

**dispose(auxa);**

**resultado:=true end**

**//tiene hijo izq**

**else if ((a^.HI <> nil) and (a^.HD = nil) ) then begin**

**auxa:=a;**

**a:=a^.HI;**

**dispose(auxa);**

**resultado:=true end**

**//SI TIENE DOS HIJOS**

**else begin**

**auxa:=buscar(a,dato,encontro);**

**auxa^.dato:=vermin(a^.HD);**

**borrarelemento(a^.HD,auxa^.dato,resultado);**

**end**

**end;**