Taller De Programacion Unlp

Modulo Imperativo

ALGORITMOS

Emiliano Francisco 2022

Pascal

Listas

Insertar Atras

```
Procedure insertarAtras (var l: lista; d: dato);

var
  aux: lista;
  nue: lista;

Begin
   new(nue);
   nue^.dato:= d;
   nue^.sig:= Nil;
   if (l = nil) then
      l:= nue
   else
   Begin
      aux:= l;
      while (aux^.sig <> nil) Do
        aux:= aux^.sig;
      aux^.sig:= nue;
   end;
end;
```

Insertar Atras 2

```
Procedure insertarAtras2 (var Pp,Up: lista; d: dato);

var
  nue: lista;

Begin
   new(nue);
   nue^.dato:= d;
   nue^.sig:= Nil;
   if (Pp = nil) then
      Pp:= nue
   else
    Up^.sig:= nue;
   Up:= nue;
end;
```

Insertar Ordenado

```
Procedure insertarOrdenado (var l: lista; d: dato);

var
  nue,ant,act: lista;

Begin
   new(nue);
   nue^.dato:= d;
   act:= l;
   while (act <> Nil) and (dato < act^.dato) Do
   Begin
      ant:= act;
      act:= act^.sig;
   end;
   if (l = act) then
      l := nue
   else
      ant^.sig:= nue;
   nue^.sig:= act;
end;
```

Pascal

Ordenamiento de vector

Seleccion

```
Procedure Ordenar (var v: tVector; dimLog: integer);

var
i, j, p: integer;
item: integer;

Begin
   for i:=1 to dimLog - 1 Do
   Begin
      p := i;
      for j:= i + 1 to dimLog Do
      if v[j] < v[p] Then
         p:= j;
      item := v[p];
      v[p] := v[i];
      v[i] := item;
   end;
end;
```

Insercion

```
Procedure Ordenar (var v: tVector; dimLog: integer);

var
i, j, p: integer;
actual: integer;

Begin
   for i:= 2 to dimLog Do
   Begin
      j:= i-1;
      while (j>0) and (v[j]>actual) Do
      Begin
         v[j+1]:= v[j];
         j:= j - 1;
      end;
      v[j+1]:= actual;
    end;
end;
```

Pascal

Recursividad

Funcion / Suma Elementos de Lista

```
Function SumaElementosLista (Pp: lista): integer;

Begin
  If (Pp = Nil) Then
    SumaElementosLista := 0
  Else
    SumaElementosLista := SumaElementosLista(Pp^.sig) + Pp^.int;
End;
```

Funcion / Suma Elementos de Arbol

```
Function sumarElementos (a: arbol): integer;

Begin
  If (a = Nil)Then
    sumarElementos := 0
  Else
    sumarElementos := a^.int +sumarElementos(a^.hi) +
    sumarElementos(a^.hd);
End;
```

Funcion / Max-Min Arbol

```
Function calcularMin (a:arbol): integer;

Begin
  If (a^.hi = Nil) Then
    calcularMin := a^.int
  Else
    calcularMin := calcularMin(a^.hi);
End;


Function calcularMax (a:arbol): integer;

Begin
  If (a^.hd = Nil) Then
    calcularMax := a^.int
  Else
    calcularMax := calcularMax(a^.hd);
End;
```

Pascal

Busqueda Dicotomica

Busqueda Dicotomica

```
Procedure busquedaDicotomica (v: vector; dimLog:integer);

Procedure busquedaDicRecursiva (v:vector; pri,ult,dato: integer;
          var pos:integer);
Begin
   pos:= ((pri + ult)Div 2);
   If ((dato = v[pos].dato) Or (pri > ult)) Then
   Begin
      If (dato = v[pos].dato) Then
         Writeln (’Dato :’, v[pos].dato)
      Else
         Writeln (’Dato no encontrado’);
   end
   Else
      If (dato < v[pos].dato) Then
      Begin
         ult:= pos - 1;
         busquedaDicRecursiva (v: vector; pri,ult,dato: integer;
         var pos:integer);
      end
      Else
      Begin
         pri:= pos + 1;
         busquedaDicRecursiva (v: vector; pri,ult,dato: integer;
         var pos:integer);
      end;
end;

Var
   pri,ult,pos,dato: integer;

Begin
   pos:= 0;
   pri:= 1;
   ult:= dimLog;
   Writeln (’Ingrese dato a buscar’);
   Readln(dato);
   busquedaDicRecursiva(v,pri,ult,dato,pos);
End;
```

Pascal

Arboles

Cargar

```
Procedure cargarArbol (Var a: arbol; x: integer);

Var
  nue: arbol;
Begin;
  If (a = Nil) Then
    Begin;
      new (nue);
      nue^.dato := x;
      nue^.hd := Nil;
      nue^.hi := Nil;
      a := nue;
    End
  Else
    Begin;
      If (a^.dato < x) Then
        cargarArbol (a^.hd, x)
      Else
        cargarArbol (a^.hi, x);
    End;
End;
```

Funcion / Buscar

```
Function busqueda (a: arbol; x: integer): integer;
Begin;
  If (a = Nil) Then
    busqueda := -1
  Else
    Begin;
      If (a^.dato = x) Then
        busqueda := x
      Else
        Begin;
          If (a^.dato < x) Then
            busqueda := busqueda (a^.hd, x)
          Else
            busqueda := busqueda (a^.hi, x);
        End;
    End;
End;
```

Imprimir en orden

```
Procedure imprimir (a: arbol);
Begin;
  If (a <> Nil) Then
    Begin;
      imprimir (a^.hd);
      writeln (a^.dato);
      imprimir (a^.hi);
    End;
End;
```

Pascal

Arbol de lista

Programa Completo

```
Program ArbolDeLista;

Type

  arbol = ^nodoB;

  lista = ^nodoA;

  dato = Record
    nombre : String;
    edad : integer;
    turno : integer;
  End;

  nodoA = Record
    d: dato;
    sig : lista;
  End;

  nodoB = Record
    hi: arbol;
    hd: arbol;
    l: lista;
    turno: integer;
  End;

Procedure agregarNodo (Var l:lista; d: dato);

Var
  nue, ant, act: lista;
Begin
  new(nue);
  nue^.d := d;
  act := l;
  While (act <> Nil) And (d.edad > act^.d.edad)  Do
    Begin
      ant := act;
      act := act^.sig;
    End;
  If (act = l) Then
    l := nue
  Else
    ant^.sig := nue;
  nue^.sig := act;
End;

Procedure agregarHoja (Var a: arbol; d: dato);

Var
  nue: arbol;
  nL: lista;
Begin
  If (a = Nil) Then
    Begin
      new(nue);
      nue^.hi := Nil;
      nue^.hd := Nil;
      nue^.turno := d.turno;
      new(nl);
      nl^.d := d;
      nl^.sig := Nil;
      nue^.l := nL;
      a := nue;
    End
  Else
    If (d.turno = a^.turno) Then
      agregarNodo(a^.l,d)
  Else
    If (d.turno < a^.turno) Then
      agregarHoja(a^.hi,d)
  Else
    agregarHoja(a^.hd,d)
End;

Procedure cargarDato (Var a:arbol);

Var
  d: dato;
Begin
  WriteLn ('Ingrese turno');
  ReadLn (d.turno);
  While (d.turno <> 0) Do
    Begin
      WriteLn ('Ingrese nombre');
      ReadLn (d.nombre);
      WriteLn ('Ingrese edad');
      ReadLn (d.edad);
      agregarHoja(a,d);
      WriteLn ('Ingrese turno');
      ReadLn (d.turno);
    End;
End;

Procedure mostrarDato (l: lista);

Var
  aux: lista;
Begin
  If (l <> Nil) Then
    aux := l;
  While (aux <> Nil) Do
    Begin
      WriteLn('Nombre: ',aux^.d.nombre);
      WriteLn('Edad: ',aux^.d.edad);
      aux := aux^.sig;
    End;
End;

Procedure mostrarDatos (a:arbol);
Begin
  If (a <> Nil) Then
    Begin
      mostrarDatos(a^.hi);
      WriteLn('Turno: ',a^.l^.d.turno);
      mostrarDato(a^.l);
      mostrarDatos(a^.hd)
    End;
End;

Var
  a: arbol;
  opt: integer;

Begin
  Repeat
    WriteLn ('1- Cargar Datos');
    WriteLn ('2- Mostrar Datos');
    WriteLn ('0- Salir');
    ReadLn(opt);
    Case opt Of
      1: cargarDato(a);
      2: mostrarDatos(a);
    End;
  Until (opt = 0);
End.
```

Pascal

Merge

Merge

cut:= 999 (variable global)

```
Procedure Merge (v:vector; var Pp,Up: lista2);

Procedure Minimo (var v:vector; var min: dato2);

Var
   pos: integer;
   minPos: integer;

Begin
   min.code := cut;
   For pos:= 1 to 7 Do
   Begin
      If ((v[ pos] <> Nil) and (v[pos]^.dato.cod < min.cod)) Then
      Begin
         minPos:= pos;
         min.cod:= v[pos]^.dato.cod;
      end;
   end;
   If (min.cod <> cut) Then
      v[minPos] := v[minPos]^.sig;
End;

Var
   min:= dato2;
   act:= sub;
   tot:= integer;

Begin
   min.cod := 0;
   min.total := 0;
   Minimo(v,min);
   While (min.cod <> cut) Do
   Begin
      act := min.cod;
      tot := 0;
      While ((min.cod <> cut) And (act = min.cod)) Do
      Begin
         tot := tot + 1;
         Minimo(v,min);
      end;
      min.total := tot;
      agregarAtras(Pp,Up,min);
   end;
End;
```