Redictado Taller de programación 2020

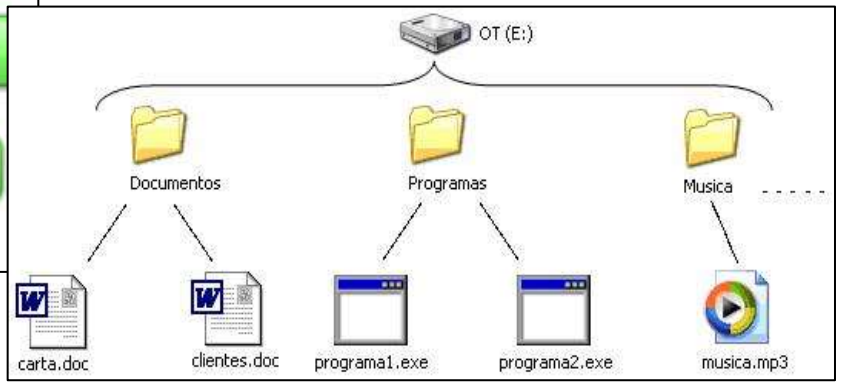
CLASE 3

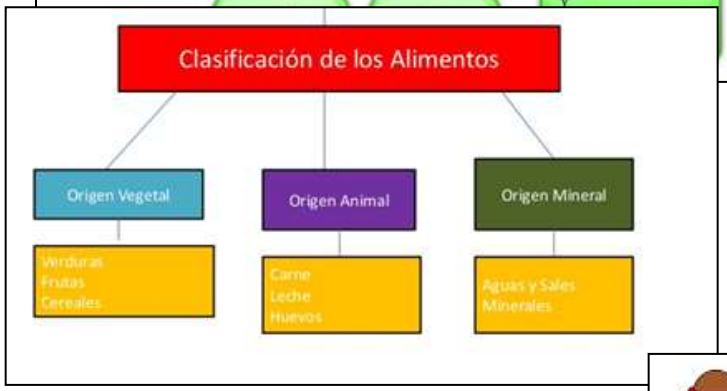
Recursión - Árboles

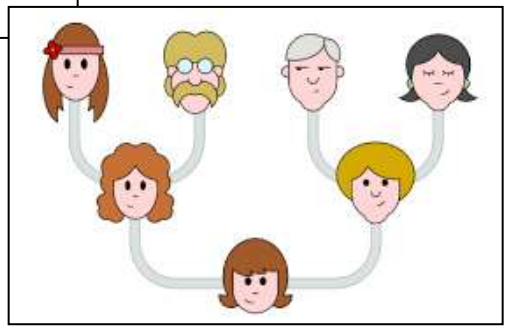
Temas de la clase

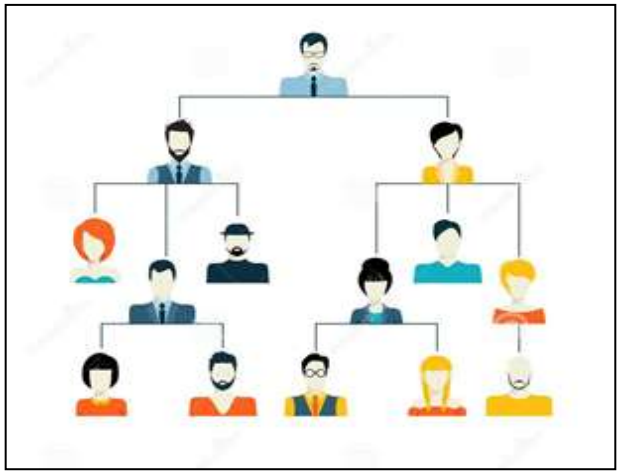
- Caso de uso de Recursión: Árbol Binario
- Árboles Binarios. Definición y características.
- Operaciones con Árboles Binarios de Búsqueda


ANIMALES
VERTEBRADOS
INVERTEBRADOS
MAMÍFEROS
AVES
PECES
PORÍFEROS
GUSANOS
MOLUSCOS
ARTRÓPODOS
EQUINODERMOS


OT (E:)
Documentos
Programas
Musica
carta.doc
clientes.doc
programa1.exe
programa2.exe
musica.mp3


Clasificación de los Alimentos
Origen Vegetal
Origen Animal
Origen Mineral
Verduras
Frutas
Cereales
Carne
Leche
Huevos
Aguas y Sales
Minerales

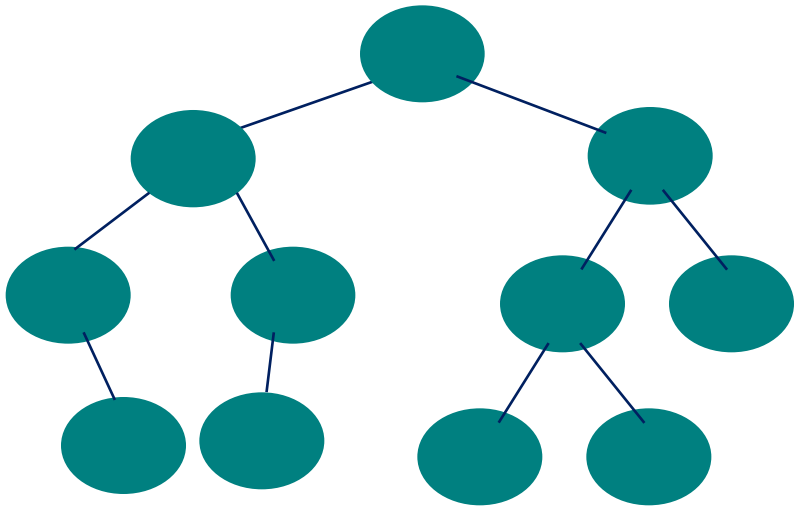




Árbol Binario – Definición y Características

Un **árbol binario** es una estructura de datos con las siguientes características:

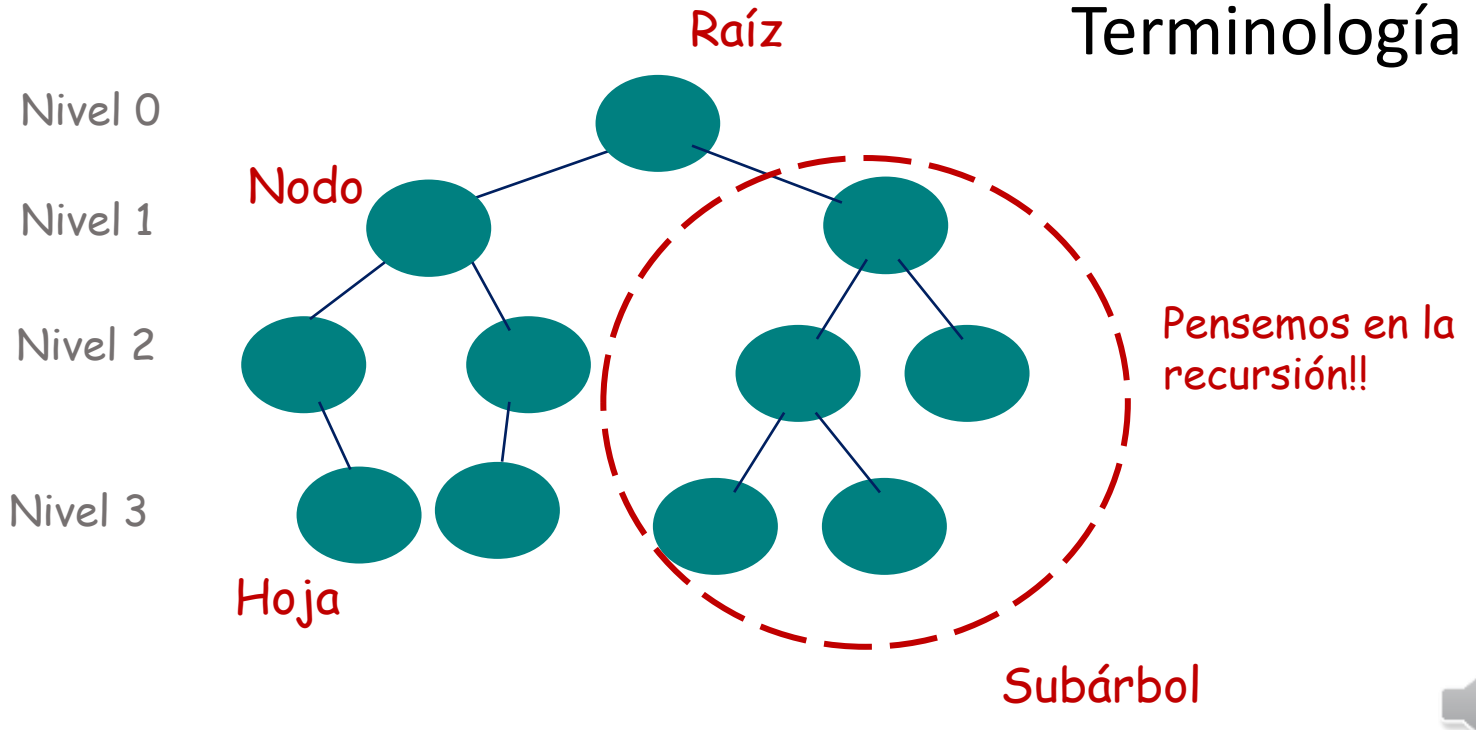
1. Homogénea: todos los elementos son del mismo tipo
2. Dinámica: puede aumentar o disminuir su tamaño durante la ejecución del programa
3. No lineal: cada elemento puede tener 0, 1, o 2 sucesores
4. Acceso Secuencial



Árbol Binario – Definición y Características

- Cada elemento del árbol se relaciona con cero, 1 o 2 elementos (hijos).
- Si el árbol no está vacío, hay un único elemento (raíz) y que no tiene padre (predecesor).
- Todo otro elemento del árbol posee un único padre y es un descendiente de la raíz.

Terminología

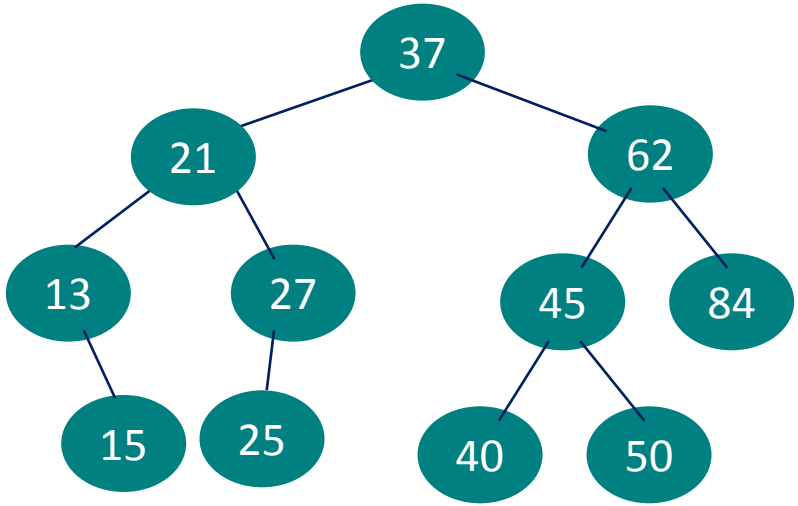


Árbol Binario de Búsqueda (ABB)

Cada nodo tiene un valor que

- Es mayor que el valor de todos los nodos del subárbol izquierdo
- Es menor que el valor de todos los nodos del subárbol derecho

Utilidad más importante ➔ Búsquedas: el tiempo medio es O(log n))



Árbol Binario - Representación

```
Type

  elemento = tipoElemento;

  arbol = ^nodo;

  nodo = record
           elem: elemento;
           hijoIzq: arbol;
           hijoDer: arbol;
         end;
```

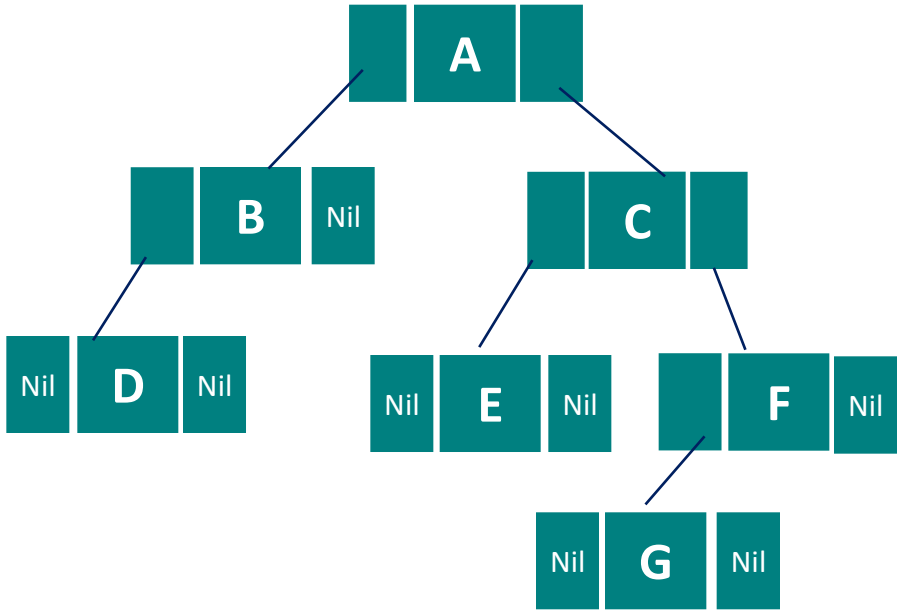


ABB – Operación Insertar un dato

Consideraciones:

- Al principio el árbol esta vacío (puntero a raíz **a** es nil)
- Siempre se inserta a nivel hoja (respetando criterio orden)
- Supongamos que se lee: 20 7 36 1 4 23

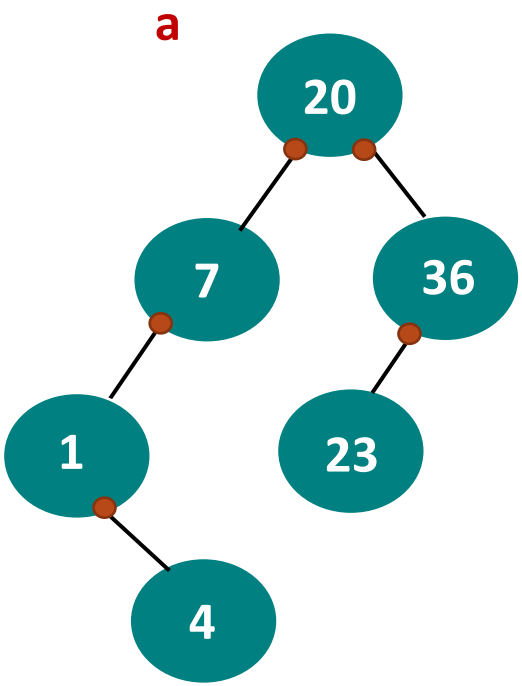
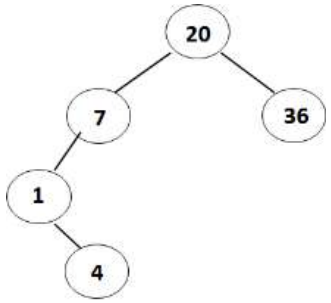


ABB – Operación Insertar un dato

```
insertar (arbol, dato)
  si arbol es nil
    creo nodo_nuevo y pongo el dato y los hijos en nil
    arbol := nodo_nuevo
  sino
    si el dato en árbol es > dato
      insertar(hijo_izquierdo_del_árbol, dato);
    sino
      insertar(hijo_derecho_del_árbol, dato);
```

¿Qué pasa si los valores a insertar estuvieran **repetidos**?



Actividades en Máquina

ACTIVIDAD 1

Descargar de Ideas ProgramaArbol.pas y realizar las siguientes tareas:

a) Analizar las declaraciones de tipos y módulos.

b) Incorporar el módulo **Insertar** que se presenta a continuación:

```
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 begin
     if (a^.dato>num) then
        insertar(a^.HI,num)
     else
        insertar(a^.HD,num);
  end;
end;
```

Actividades en Máquina

c) Implementar el módulo **CrearABB** que lee valores enteros que se ingresan por teclado (finaliza con -1) y los guarde en un árbol binario de búsqueda. Utilizar el **procedure Insertar** presentado.

d) En el programa principal, invocar al módulo **CrearABB** para generar un árbol y al módulo **ImprimirPorNivel** con el árbol resultante.

e) Ejecutar el programa con los valores: 50, 70, 30, 20, 85, 75, 78, 25, -1

f) Graficar en papel el ABB y comprobar que los datos que muestra el programa se corresponden con la estructura graficada.

ABB – Recorridos

Los distintos recorridos permiten desplazarse a través de todos los nodos del árbol de tal forma que cada nodo sea visitado una y solo una vez.

Existen varios métodos que se diferencian en el orden que se visitan los nodos:

- **Recorrido En – Orden** (subárbol izquierdo – raíz – subárbol derecho)
- **Recorrido Pre – Orden** (raíz - subárbol izquierdo – subárbol derecho)
- **Recorrido Post – Orden** (subárbol izquierdo – subárbol derecho - raíz)

```
procedure enorden(a:arbol);
begin
  if (a <> nil) begin
      enorden (a^.HI);
      write (a^.dato);
      enorden (a^.HD);
  end;
end;
```

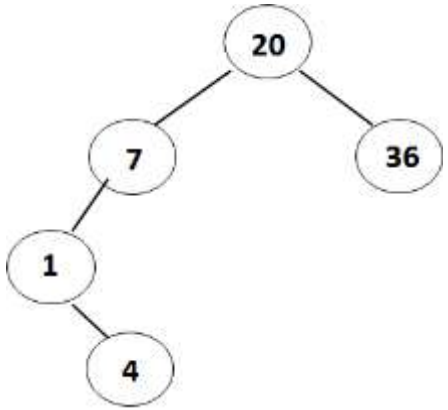
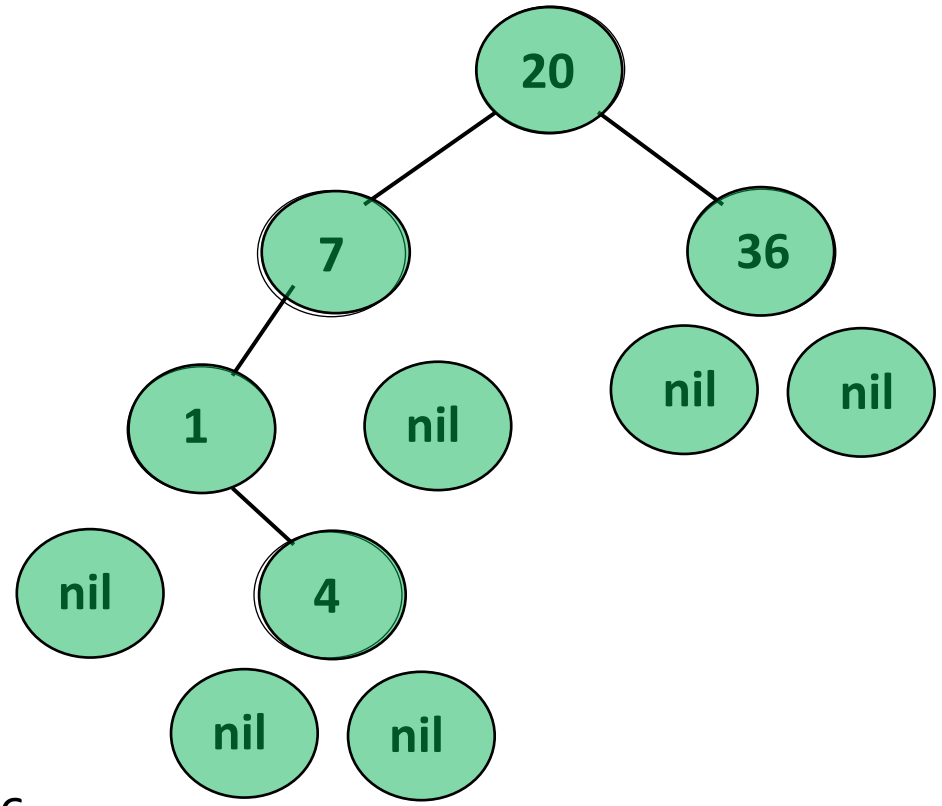


ABB – Recorridos – En Orden

```
procedure enorden(a:arbol);
begin
  if (a <> nil) begin
      enorden (a^.HI);
      write (a^.dato);
      enorden (a^.HD);
  end;
end;
```



Salida: 1 4 7 20 36

Actividades en Máquina

ACTIVIDAD 2

En el programa **ProgramaArbol.pas** realizar las siguientes tareas:

a) Implementar el módulo **preOrden** que imprima los valores del ABB ya generado.

b) Implementar el módulo **enOrden** que imprima los valores del ABB ya generado.

c) Implementar el módulo **postOrden** que imprima los valores del ABB ya generado.

d) Invocar cada uno de los módulos anteriores y comparar los resultados obtenidos.

Actividades en Máquina

ACTIVIDAD 3

En el programa **ProgramaArbol.pas** realizar las siguientes tareas:

a) Implementar el módulo **Buscar** que reciba un ABB y un valor y devuelva el puntero al nodo donde se encuentra dicho valor. En caso de no encontrarlo, debe retornar nil.

b) En el programa principal, invocar al módulo **Buscar** con un valor que se ingresa de teclado. Informar si el valor buscado se encontró en el árbol o no.

```
function Buscar(a:arbol;d:integer):arbol;
```

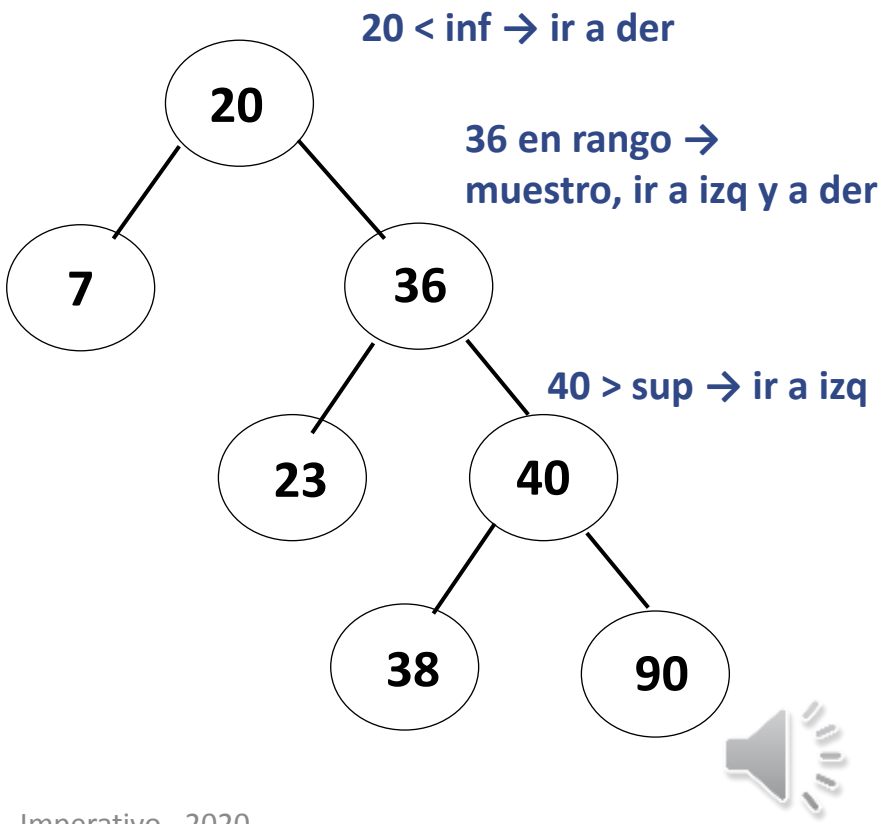
ABB – Recorridos Acotado

Algunas situaciones nos obligan a recorrer todos los nodos del árbol, por ejemplo imprimir los datos del árbol

Y si necesitamos mostrar los datos que están comprendidos entre dos valores determinados del árbol ¿cómo lo resolvemos?

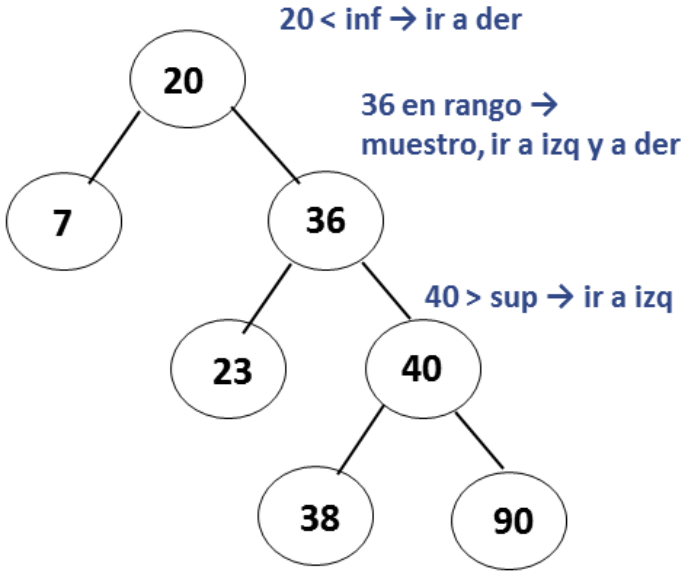
Ej. Datos e/ 21 y 39

```
ImprimirAcotado(arbol, inf, sup);
  si arbol no está vacío
     si el valor en arbol es >= inf and
          el valor en arbol es <= sup
               mostrar valor
     ImprimirAcotado (hijo_izq_arbol, inf, sup)
     ImprimirAcotado (hijo_der_arbol, inf, sup)
```



¿es eficiente? NO

ABB – Recorridos Acotado




```
recorridoAcotado(arbol, inf, sup);

si arbol no está vacío
  si el valor en arbol es >= inf
    si el valor en arbol es <= sup
        mostrar valor
        recorridoAcotado (hijo_izq_arbol, inf, sup);
        recorridoAcotado (hijo_der_arbol, inf, sup);
    sino
        recorridoAcotado (hijo_izq_arbol, inf, sup);
  sino
    recorridoAcotado (hijo_der_arbol, inf, sup);
```

Actividades en Máquina

ACTIVIDAD 3

En el programa **ProgramaArbol.pas** realizar las siguientes tareas:

c) Implementar el módulo **VerValoresEnRango** que reciba un ABB y dos valores, que indiquen un rango, e informe los valores almacenados que se encuentren en dicho rango.

d) En el programa principal, invocar al módulo **VerValoresEnRango** con dos valores leídos de teclado.

```
procedure VerValoresEnRango(a:arbol; inf,sup: integer);
```

Actividades en Máquina

ACTIVIDAD 4

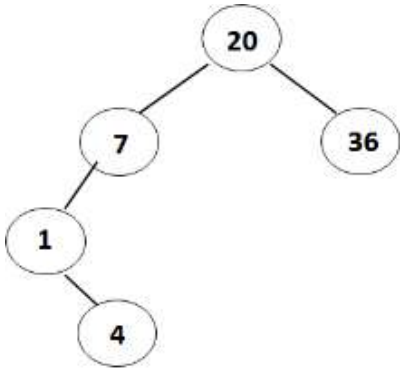
En el programa **ProgramaArbol.pas** realizar las siguientes tareas:

a) Implementar el módulo **VerMin** que reciba un árbol y devuelva el valor mínimo. En caso de recibir un árbol vacío, retornar -1.

b) Implementar el módulo **VerMax** que reciba un árbol y devuelva el valor máximo. En caso de recibir un árbol vacío, retornar -1.

c) En el programa principal, invocar a los módulos generados en a) y b). Informar los resultados obtenidos.

```
function VerMin(a:arbol): integer;
```



Enviar a través de la Mensajería de Ideas, programaArbol.pas al docente asignado al grupo.

Actividades en Máquina

ACTIVIDAD 5

Implementar un programa que procese la información de los participantes a un concurso de preguntas y respuestas. De cada participante se lee el código de participante, código de ciudad de origen y edad. El ingreso de los participantes finaliza cuando se lee el código -1.

Implementar un programa que:

a) Genere un ABB a partir de la información leída ordenado por código de participante.

b) Contenga un módulo que reciba el árbol generado en a) y un código de ciudad y retorne una lista con los participantes de esa ciudad.

c) Invoque al módulo de b) y luego muestre el contenido de la lista resultante utilizando un módulo recursivo.

d) Muestre la cantidad de participantes cuyos códigos están comprendidos entre dos valores determinados. Para ello implementar un módulo que reciba el árbol generado en a) y dos valores y devuelva la cantidad pedida.

e) Informe la edad promedio de los participantes del concurso.