


Redictado Taller de Programación 2022

CLASE 4

Recursión - Árboles

A silver laptop sits on a light-colored wooden desk. The laptop screen is white and displays four lines of black text in a monospaced font, which is Pascal code for a simple program that prints 'Hola mundo'.

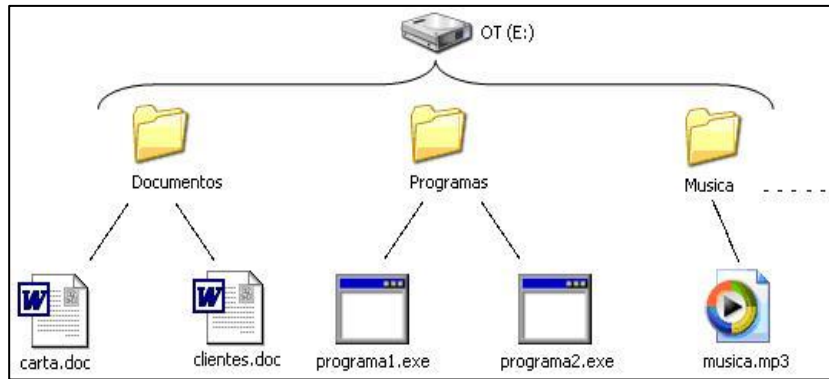
```
Program HolaMundo;  
Begin  
  writeln('Hola mundo');  
end.
```

Temas de la clase

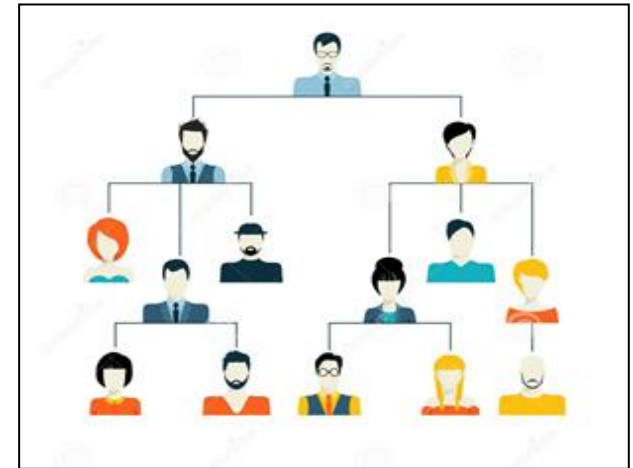
- Caso de uso de Recursión: Árbol Binario
- Árboles Binarios. Definición y características.
- Operaciones con Árboles Binarios de Búsqueda

Árbol: Estructura jerárquica. Relaciones padre-hijo.

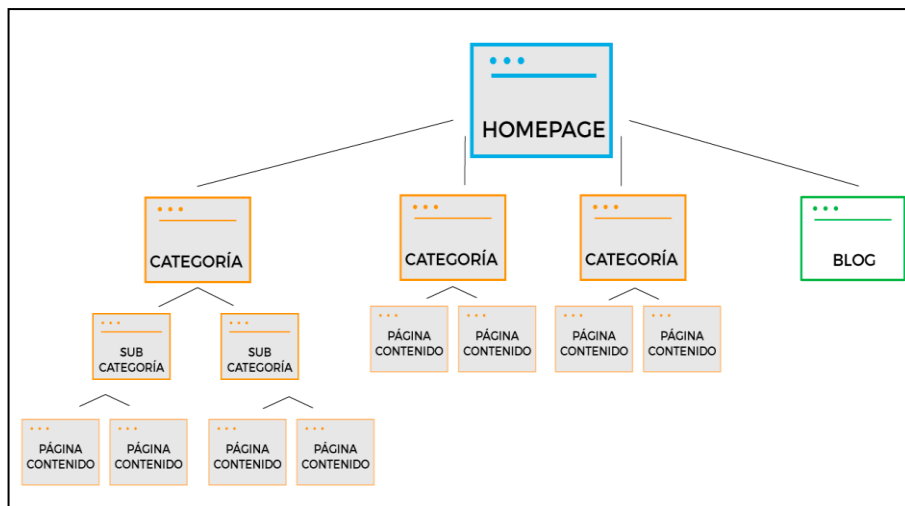
Sistemas de archivos



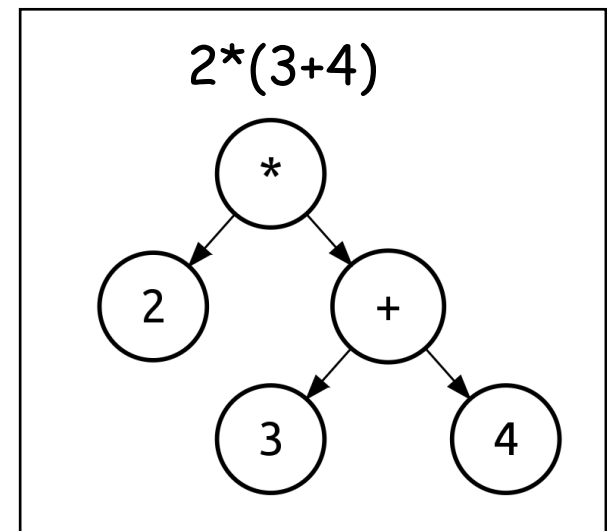
Organigrama de empresa



Estructura de un sitio web, etc ...



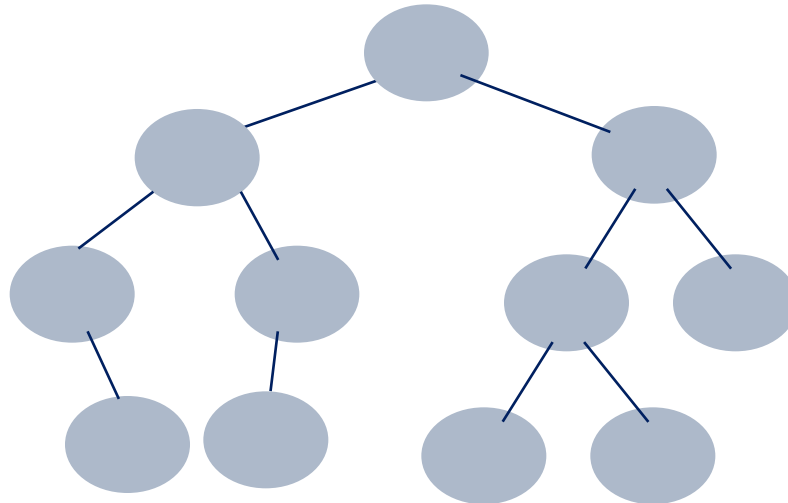
Árbol de expresión



Árbol Binario – Definición y Características

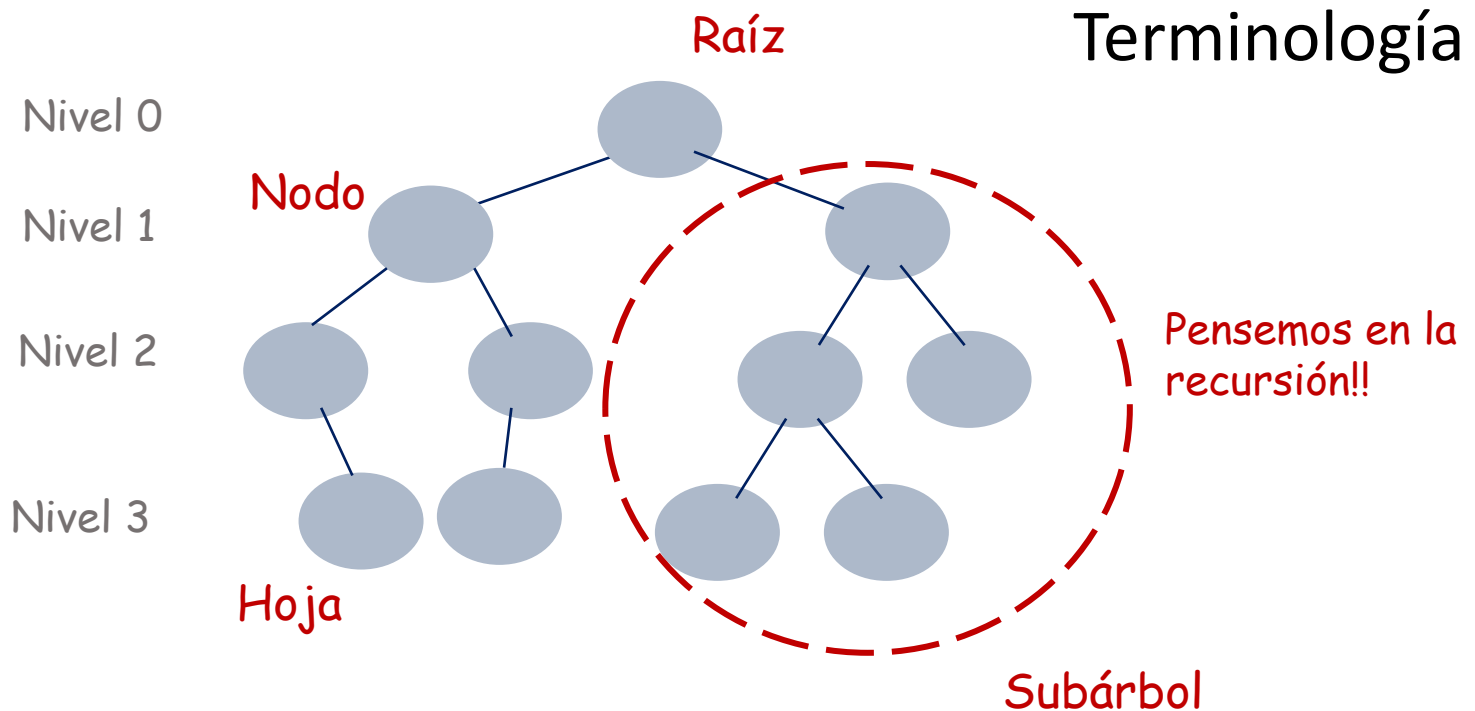
Un **árbol binario** es una estructura de datos con las siguientes características:

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.

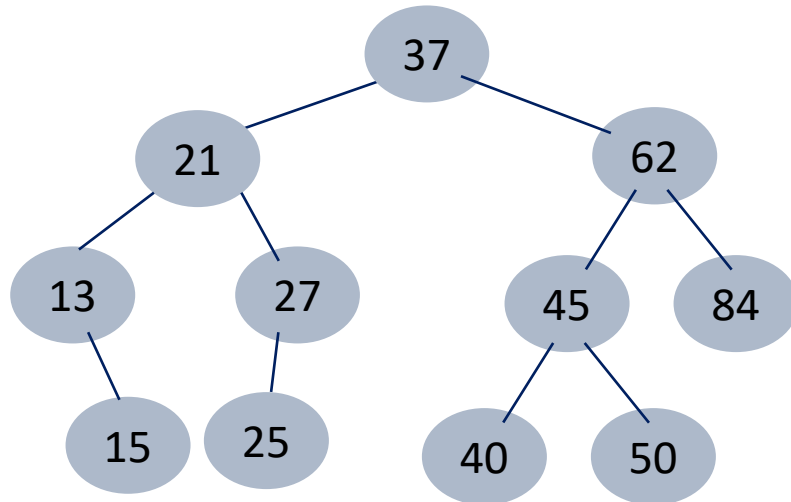


Árbol Binario de Búsqueda (ABB)

En un ABB cada nodo tiene un valor que

- Es más grande 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 eficientes: el tiempo *medio* es $O(\log n)$ si el árbol está *balanceado*.



Comparación con
listas

¿Cómo se busca
en un abb?

Árbol Binario - Representación

Type

```
tipoDato= ...;
```

```
arbol = ^nodo;
```

nodo = record

```
dato: tipoDato;
```

```
hijoIzq: arbol;
```

```
hijoDer: arbol;
```

end;

Var

```
a: arbol; {puntero a raíz}
```

Begin

• • •

End.

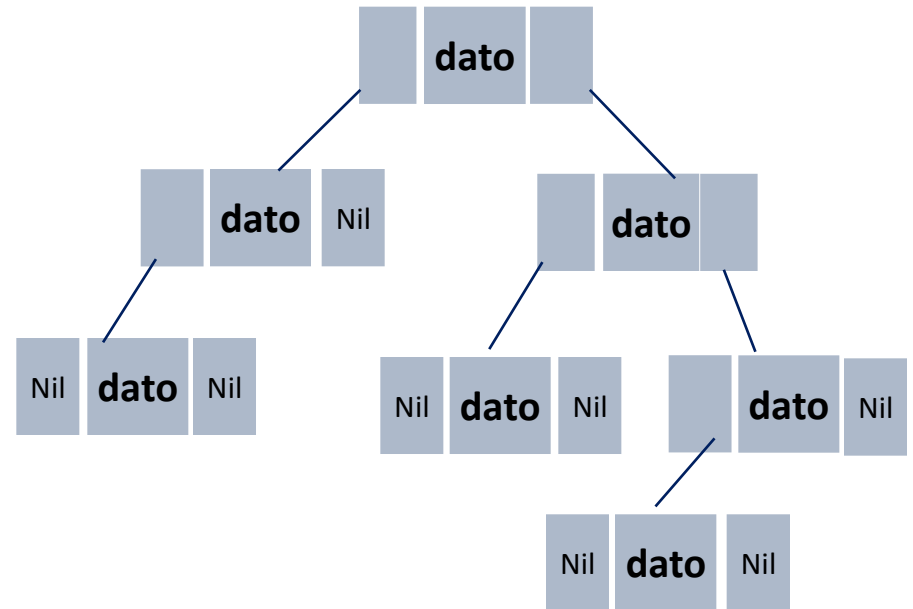
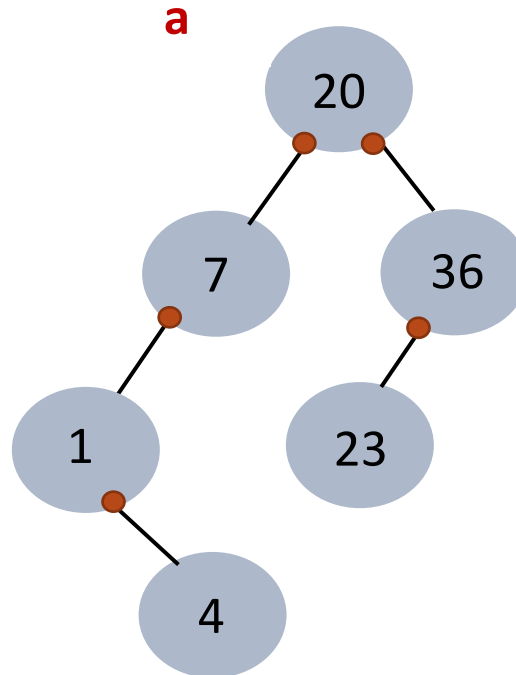


ABB – Operación Insertar un dato

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



¿Qué pasa con el abb
si los valores leídos
están ordenados?

ABB – Operación Insertar un dato - PSEUDOCÓDIGO

```
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);
```

ABB – Operación Insertar un dato - CÓDIGO

Type

```
arbol= ^nodoA;  
nodoA = Record  
  dato: integer;  
  HI: arbol;  
  HD: arbol;  
End;
```

```
procedure Insertar(var a: arbol; dato: integer);  
begin
```

```
  if (a = nil) then begin  
    new(a);  
    a^.dato:= dato;  
    a^.HI:= nil;  
    a^.HD:= nil;
```

```
  end
```

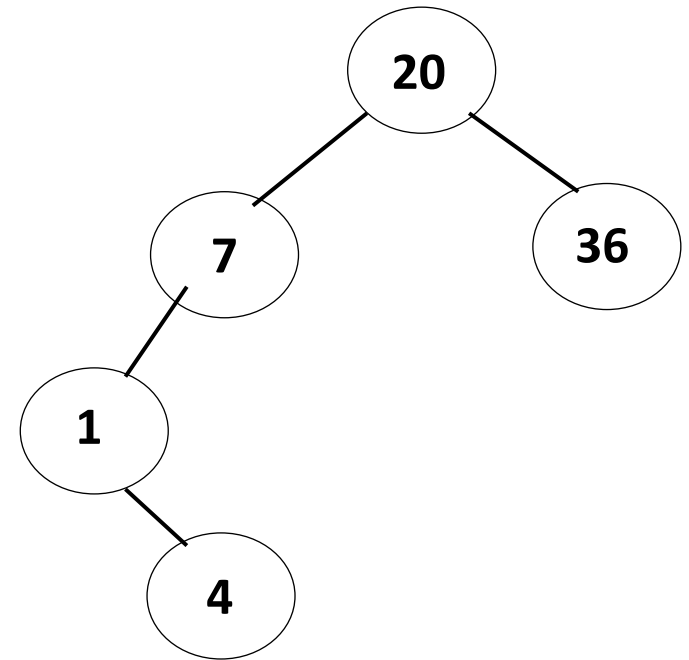
```
  else
```

```
    if (a^.dato > dato) then  
      Insertar(a^.HI, dato)
```

```
    else
```

```
      Insertar (a^.HD, dato);
```

```
end;
```



¿Cómo se inserta el 23?

¿Cómo se inserta el 1?

¿Cómo evitaría insertar repetidos?

¿Cómo contaría las apariciones de cada valor?



Actividades en Máquina

ACTIVIDAD 1

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

- a) Analizar las declaraciones de tipos y módulos.
- b) Copiar el módulo **Insertar** y modificarlo para que no almacene valores repetidos.
- c) Implementar el módulo **CrearABB** que lea valores enteros que se ingresan por teclado (finaliza con 0) y los inserte en un árbol binario de búsqueda.
- 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: 20 7 36 1 4 23 1 0
- f) Comprobar que los datos que muestra el programa se corresponden con la estructura esperada.

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)

```
procedure enorden(a:arbol);  
begin  
  if (a <> nil) then begin  
    1. enorden (a^.HI);  
    2. //procesar a^.dato  
    3. enorden (a^.HD);  
  end;  
end;
```

Recorrido Pre–Orden

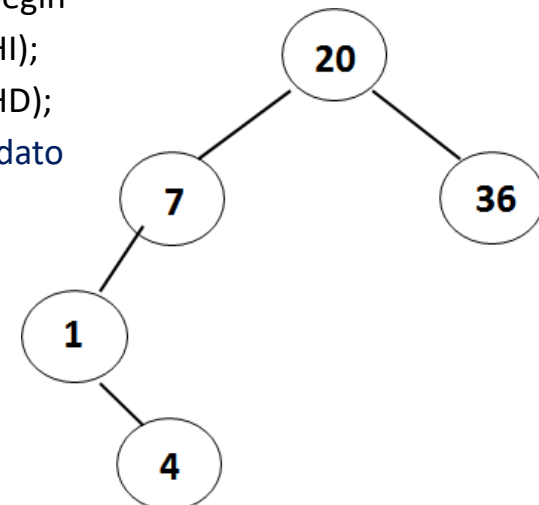
(raíz;
subárbol izquierdo;
subárbol derecho)

```
procedure preorden(a:arbol);  
begin  
  if (a <> nil) then begin  
    1. //procesar a^.dato  
    2. preorden(a^.HI);  
    3. preorden(a^.HD);  
  end;  
end;
```

Recorrido Post–Orden

(subárbol izquierdo;
subárbol derecho ;
raíz)

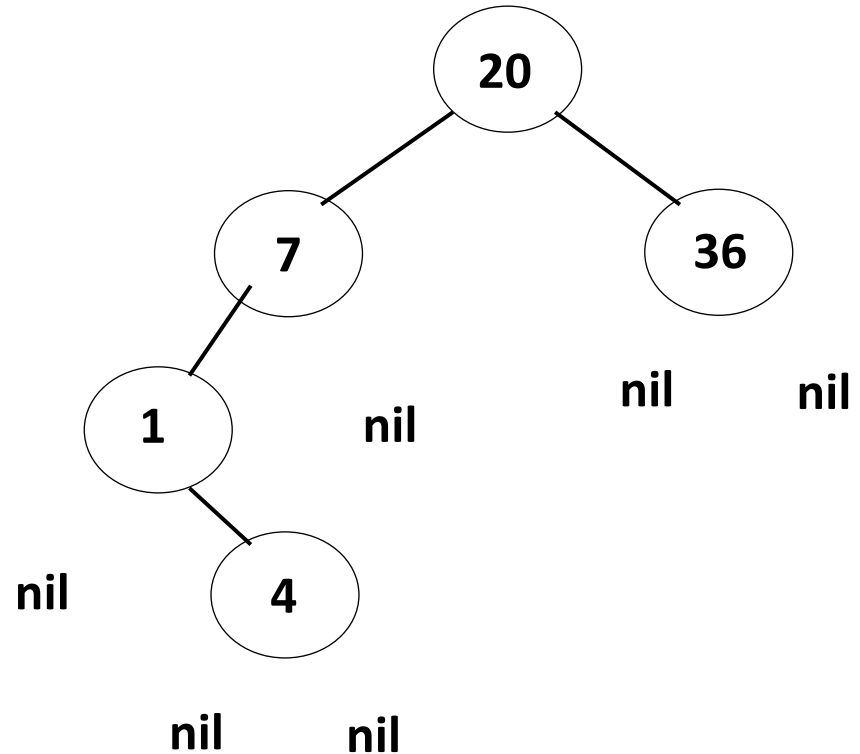
```
procedure postorden(a:arbol);  
begin  
  if (a <> nil) then begin  
    1. postorden(a^.HI);  
    2. postorden(a^.HD);  
    3. //procesar a^.dato  
  end;  
end;
```



¿Cuál elijo? Ejemplos

ABB – Recorridos – Pre Orden

```
procedure preorden(a:arbol);  
begin  
  if (a <> nil) then begin  
    1. write (a^.dato);  
    2. preorden(a^.HI);  
    3. preorden(a^.HD);  
  end;  
end;
```



¿Qué imprime?

ABB – Buscar

Permite buscar un elemento considerando el criterio de orden del árbol, retornando su puntero o nil si no existe.

¿Cómo busco el 4?

Comparamos el valor buscado con la raíz.

Si no coincide, aplicar el *mismo proceso*:

- al subárbol izquierdo, si el buscado es menor
- al subárbol derecho, si el buscado es mayor

La búsqueda finaliza al encontrar el valor buscado o llegar a un subárbol vacío (no existía el valor).

La búsqueda en un ABB es en general, más rápida que la búsqueda secuencial en un vector o una lista.

¿Cómo busco el 23?

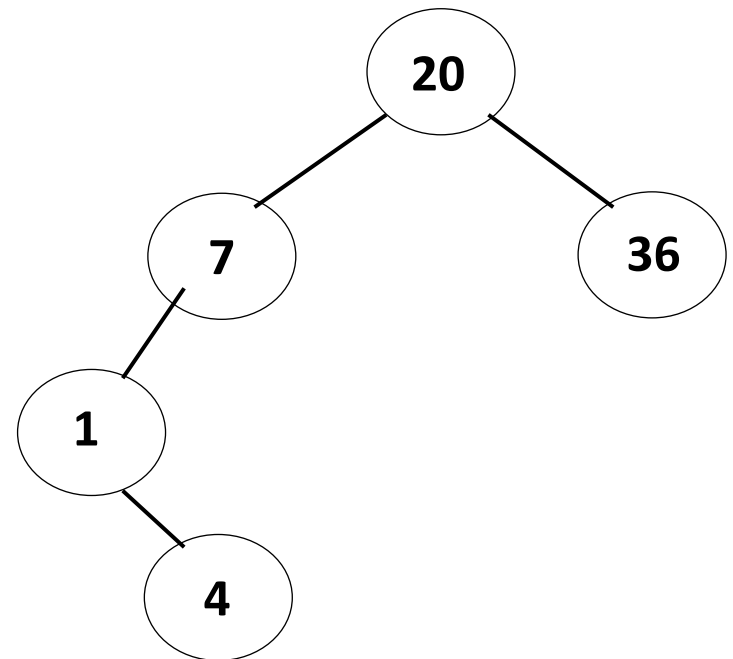


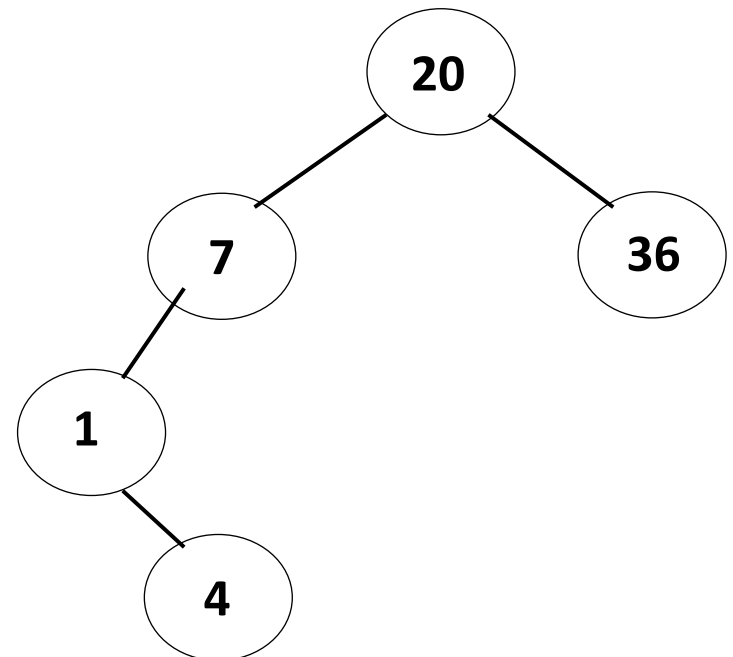
ABB – Buscar

Permite buscar un elemento considerando el criterio de orden del árbol, retornando su puntero o nil si no existe.

¿Cómo busco el 4?

¿Cómo busco el 23?

```
function Buscar (a:arbol; dato: integer):arbol;  
begin  
  if (a=nil) then  
    Buscar:=nil  
  else  
    if (dato= a^.dato) then Buscar:=a  
    else  
      if (dato < a^.dato) then  
        Buscar:=Buscar(a^.HI ,dato)  
      else  
        Buscar:=Buscar(a^.HD ,dato);  
      end;  
    end;  
end;
```



¿Cuáles son los casos base?



Actividades en Máquina

ACTIVIDAD 2

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

- a) Implementar el módulo **enOrden** que imprima los valores del ABB ya generado en orden ascendente al criterio de orden. *Pensar: ¿y si se pidiera orden descendente?*
- b) Implementar el módulo **preOrden** 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) Copiar el módulo **Buscar** que recibe un ABB y un valor y devuelve un puntero al nodo donde se encuentra dicho valor. En caso de no encontrarlo, retorna 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.



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.

Pensar: dónde se encuentra el mínimo según el criterio de orden

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

Pensar: dónde se encuentra el máximo según el criterio de orden

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



Actividades en Máquina

ACTIVIDAD 5

Implementar un programa que procese la información de los participantes de un concurso. De cada participante se lee: código de participante, código de ciudad de origen y edad. La lectura finaliza con el código de ciudad -1.

Implementar un programa que:

- a) Genere una estructura a partir de la información leída. La estructura debe ser eficiente para la búsqueda por código de participante.

A partir de la estructura generada:

- a) Genere una lista con los participantes de la ciudad cuyo código se recibe. Luego, muestre el contenido de la lista.
- b) Calcule e informe la edad promedio de los participantes del concurso.
- c) Calcule e informe el participante de menor edad.
- d) Dado un código, informe si el mismo corresponde al de un participante inscripto.
- e) Calcule e informe la cantidad de participantes cuyos códigos están comprendidos entre dos valores determinados que se reciben.