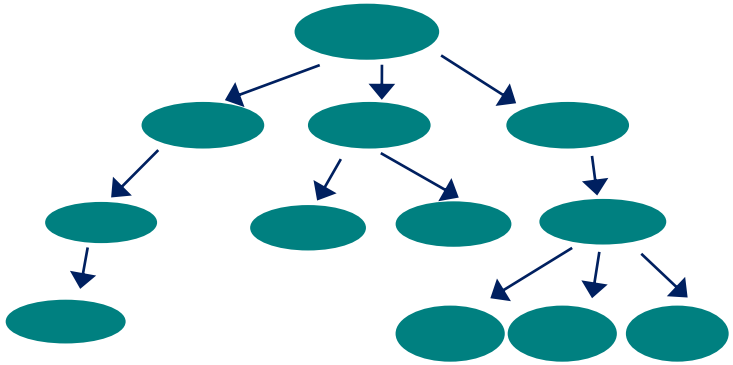
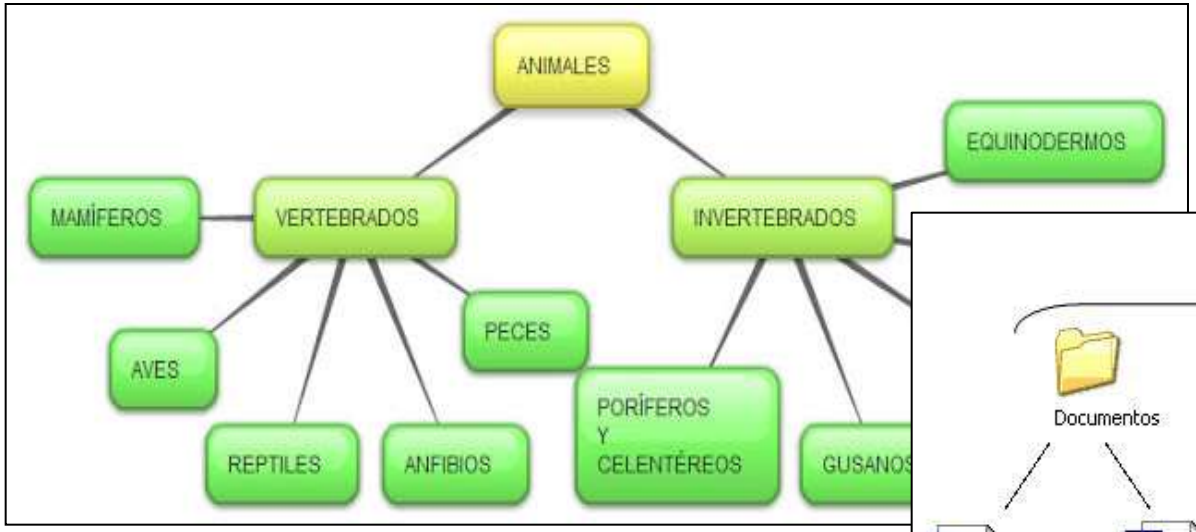
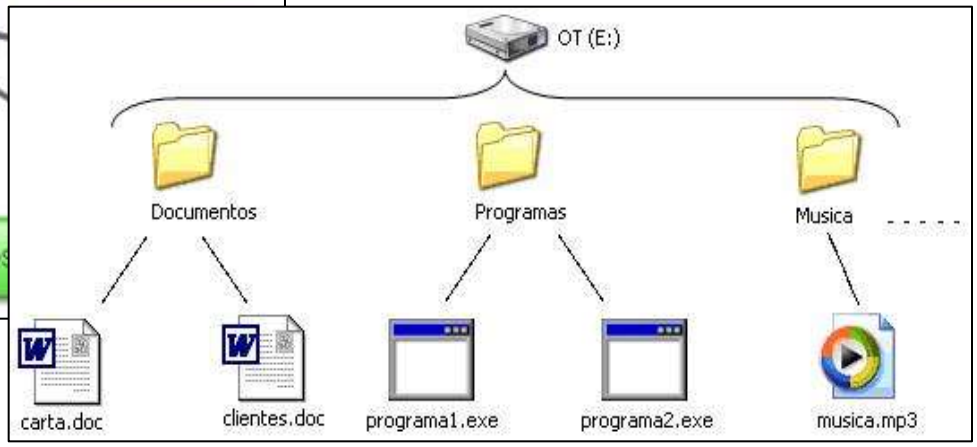


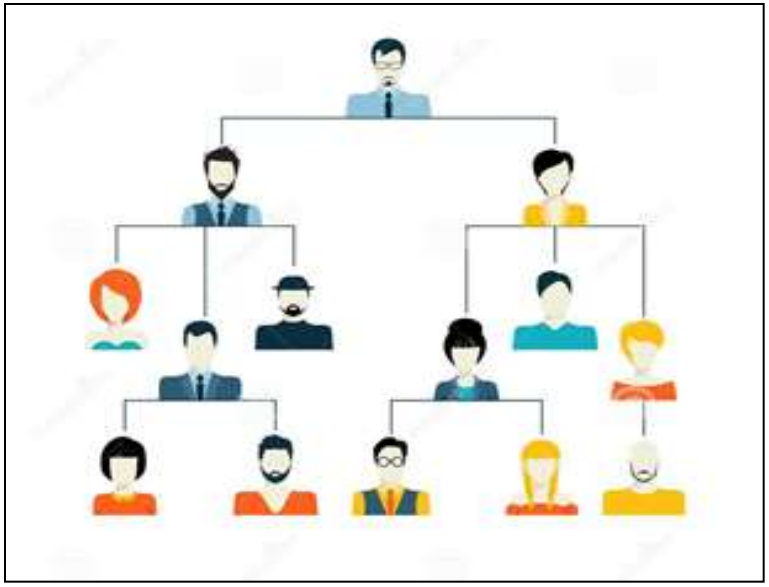
Arboles

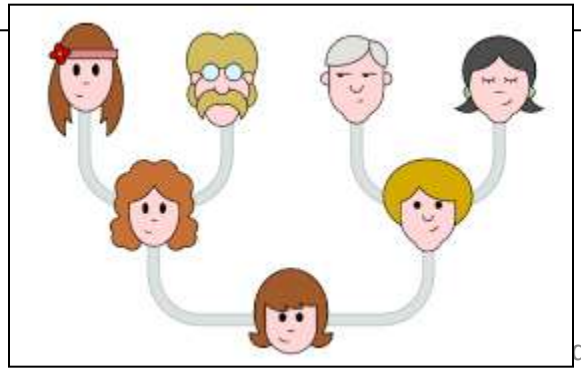
Ejemplos


ANIMALES
MAMÍFEROS
VERTEBRADOS
INVERTEBRADOS
EQUINODERMOS
AVES
PECES
REPTILES
ANFIBIOS
PORÍFEROS
Y
CELENTÈREOS
GUSANOS


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

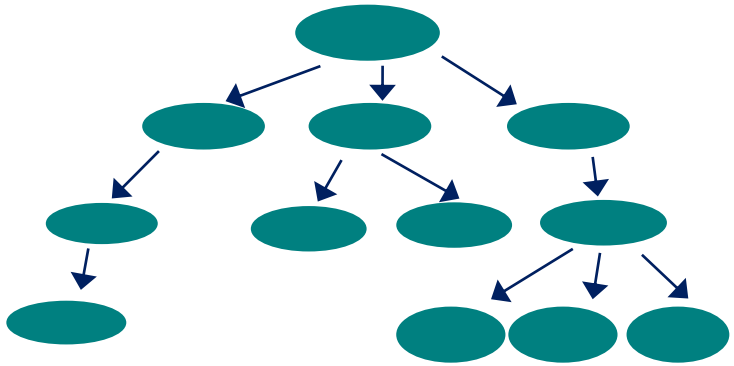

Clasificación de los Alimentos
Origen Vegetal
Origen Animal
Origen Mineral
Verduras
Frutas
Cereales
Carne
Leche
Huevos
Aguas y Sales
Minerales





Temas de la clase

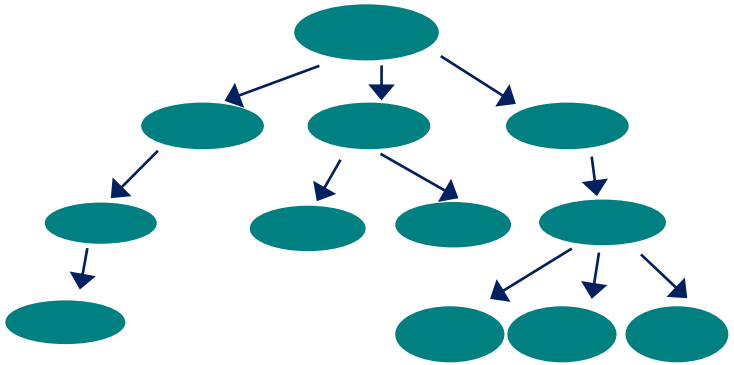
1. Árboles. Definición y características
2. Operaciones con árboles



ARBOLES

Un árbol es una estructura de datos que satisface tres propiedades:

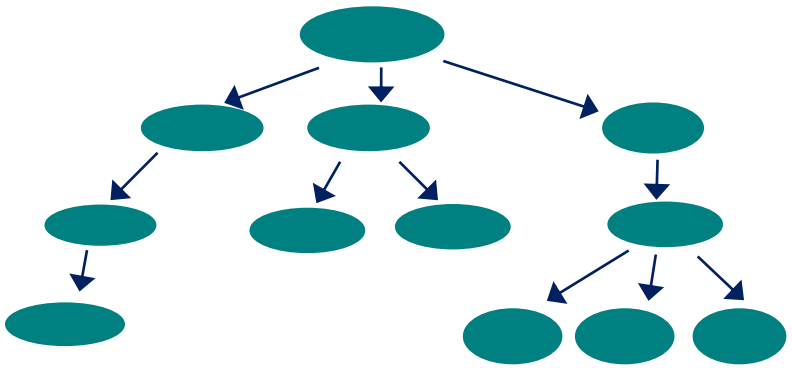
- Cada elemento del árbol se relaciona con cero o más elementos (hijos).



- Si el árbol no está vacío, hay un único elemento (raíz) y que no tiene padre (predecesor).

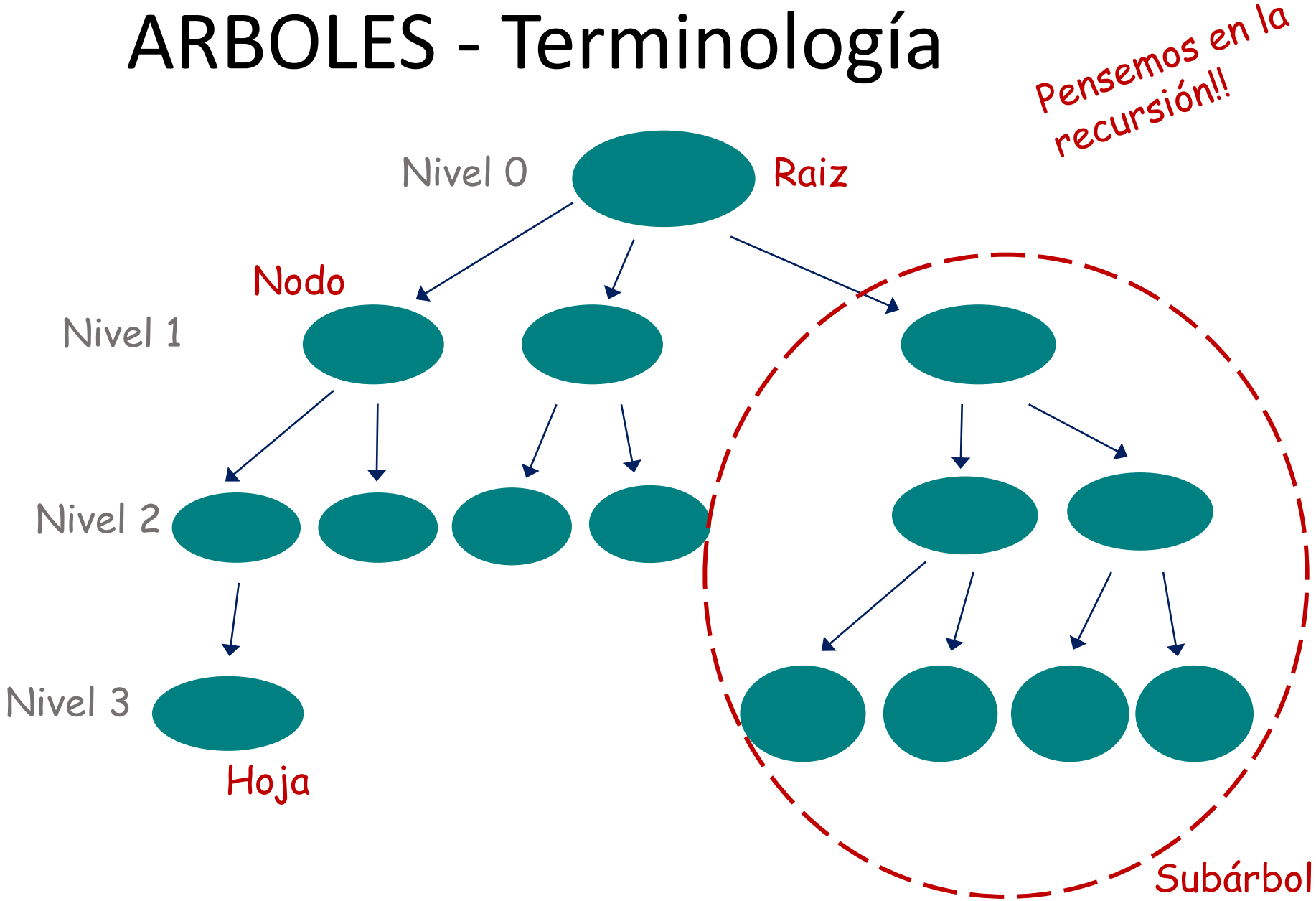
- Todo otro elemento del árbol posee un único padre y es un descendiente de la raíz.

ARBOLES - 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 más sucesores
4. Acceso Secuencial

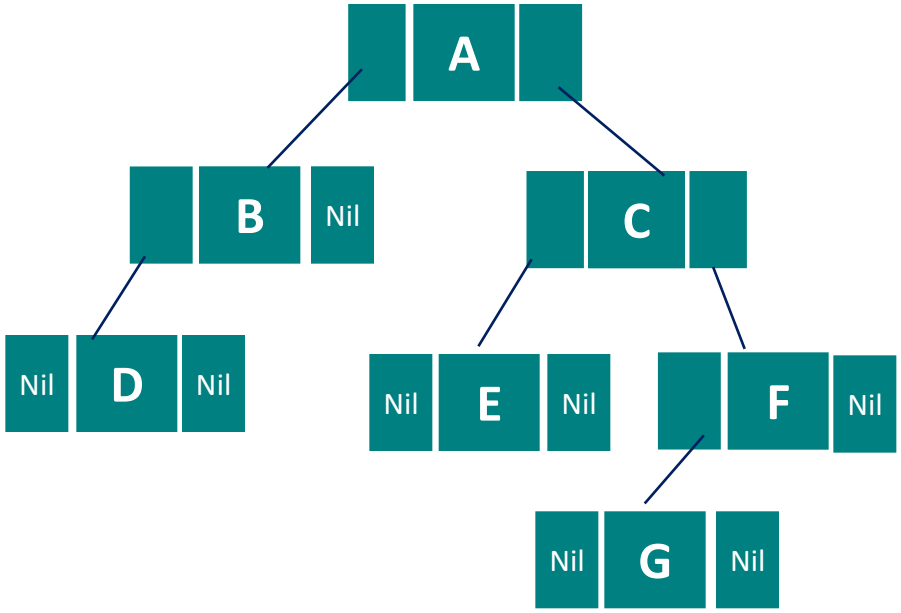
ARBOLES - Terminología



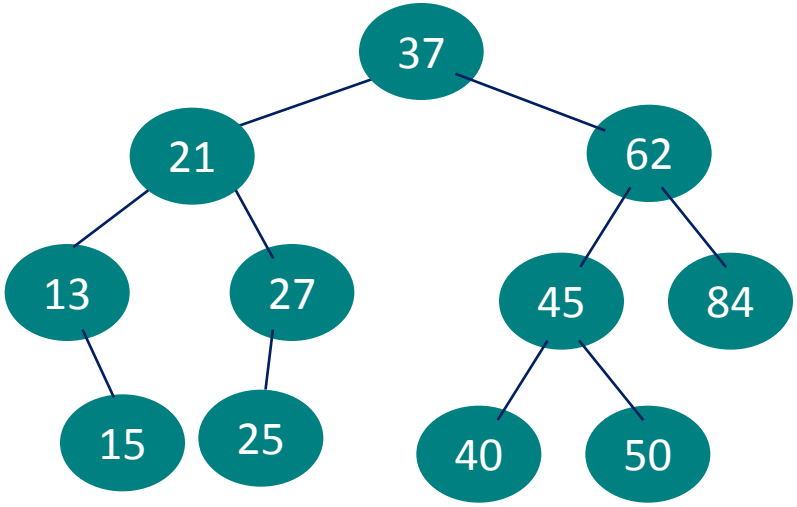
ARBOLES BINARIOS

¿Cómo se relacionan los nodos de un árbol binario?

```
Type

  elemento = tipoElemento;

  arbol = ^nodo;

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



Arboles Binarios de Búsqueda (ABB)



Observaciones

1. **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

2. **Utilidad más importante** ➔ Búsquedas el tiempo medio es O(log n))

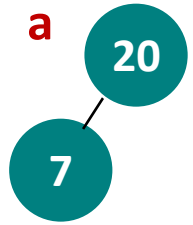
ABB – Operación Insertar Nodo

Supongamos que se leen los números: 20 7 36 1 4 23

Siempre se agrega a nivel de hoja

20 Inicialmente **a** es nil.
El nuevo nodo se convierte en la raíz del árbol. HI y HD en Nil.

7 **a** no es Nil. Debe ubicarse donde insertar.
Como 7 < 20 se toma el subárbol izquierdo de 20. Como el HI de 20 es Nil, se inserta.



36 **a** no es Nil. Debe buscar el lugar donde insertar.
Como 36 > 20 se toma el subárbol derecho. Como el HD de 20 es Nil se inserta.

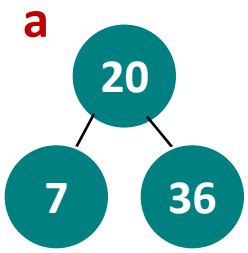
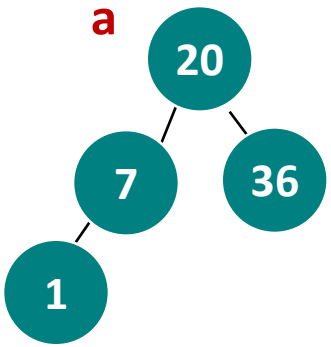


ABB – Operación Insertar Nodo

Supongamos que se leen los números: 20 7 36 1 4 23

1

a no es Nil.
Como 1 < 20 se toma el subárbol izquierdo. Como el HI no es Nil, se vuelve a comparar. Como 1 < 7, se elige el subárbol izquierdo y como es Nil se inserta.



4

a no es Nil.
Como 4 < 20 se toma el subárbol izquierdo.
Como el HI no es Nil y 4 < 7 se elige el subárbol izquierdo.
Luego, como 4 > 1, se elige el subárbol derecho y como es nil se inserta.

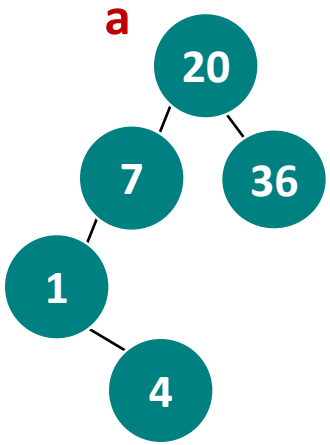
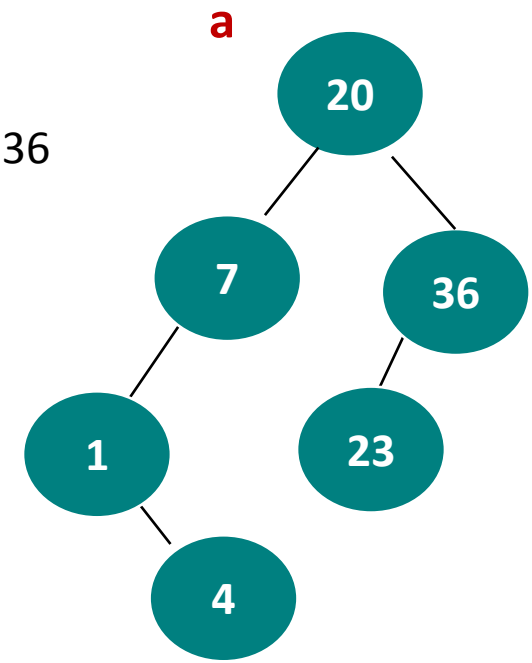


ABB – Operación Insertar Nodo

Supongamos que se leen los números: 20 7 36 1 4 23

23

a no es Nil.
Como 23 > 20 se toma el subárbol derecho.
Como el HD no es Nil se vuelve a comparar y como 23 < 36 se elige el subárbol izquierdo. Como es Nil se inserta



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?

¿Cómo podría evitar insertar valores repetidos? ¿Modificaciones en código?

Insertar un dato evitando repetidos

```
insertarSinRepetidos (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
      insertarSinRepetidos(hijo_izquierdo_del_árbol, dato);
    sino
      si el dato en el árbol < dato
         insertarSinRepetidos(hijo_derecho_del_árbol, dato);
```

Cuando el dato ya está en el árbol no realizo ninguna acción

¿Cómo haría para contar las ocurrencias de los valores repetidos?
¿Modificaciones en código?

Actividades en Máquina

ACTIVIDAD 1

Descargar de Ideas `ProgramaGenerarArbol` y realizar lo siguiente:

a) Generar una lista de números enteros (utilizar CrearListaAgregarAdelante del **`ProgramaLista`** de la clase 0)

b) Implementar el módulo **`insertar`** en un ABB de enteros

c) Invocar al módulo **`insertar`** con cada uno de los números de la lista generada anteriormente.

d) Invocar al módulo **`imprimirpornivel`** con el árbol generado en c).

e) Graficar en papel el ABB y comprobar que los datos que se muestran en d) se corresponden con la estructura generada.

Enviar el archivo `GenerarArbol` al docente asignado al grupo.

¿Qué pasaría si los valores a insertar se presentan ordenados?

Actividades en Máquina

- **ACTIVIDAD 2**

Implementar un programa que invoque a:

i) Un módulo que genere una lista ordenada de números enteros

ii) Un módulo que genere un ABB con los elementos de la lista generada.

iii) Mostrar los datos del árbol por niveles.

Nota: para generar la lista ordenada en i), reutilizar los módulos adecuados de la clase 0.

Recorridos en un árbol

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:

- Recorrido En – Orden
- Recorrido Pre – Orden
- Recorrido Post – Orden

```
Procedure preOrden( a: arbol );
begin
  if ( a <> nil ) then begin
    write (a^.dato, ' ');
    preOrden (a^.HI);
    preOrden (a^.HD)
  end;
end;
```

Actividades en Máquina

ACTIVIDAD 3

Renombrar el programa **`GenerarArbol`** como **`ProgramaArboles`** y realice 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 4

Abrir de la Medioteca de Ideas: `ProgramaEncomiendas`

`ProgramaEncomiendas` genera una lista con encomiendas.

Cada encomienda tiene código y peso. Pueden existir encomiendas de igual peso. Realice las siguientes actividades:

a) A partir de la lista, generar un árbol donde para cada peso se tengan todos los códigos de encomienda registrados con ese peso. El árbol debe estar ordenado por peso.

b) Imprimir para cada peso, los códigos de encomienda registrados para dicho peso.

Actividades en Máquina

ACTIVIDAD 5

Utilizando **`ProgramaArboles`** realice las siguientes actividades:

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

b) Invocar al módulo **`buscar`** con un valor que se ingresa de teclado. Mostrar el resultado de la búsqueda.

Actividades en Máquina

ACTIVIDAD 6

Utilizando **`ProgramaArboles`** realice las siguientes actividades:

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) Invocar a los módulos generados en a) y b). Mostrar los resultados obtenidos.