



# Ejercitación con árboles



## Ejercitación Árboles

---

### Ejemplo:

Se tiene una lista de personas anotadas en una maestría ordenada por apellido.

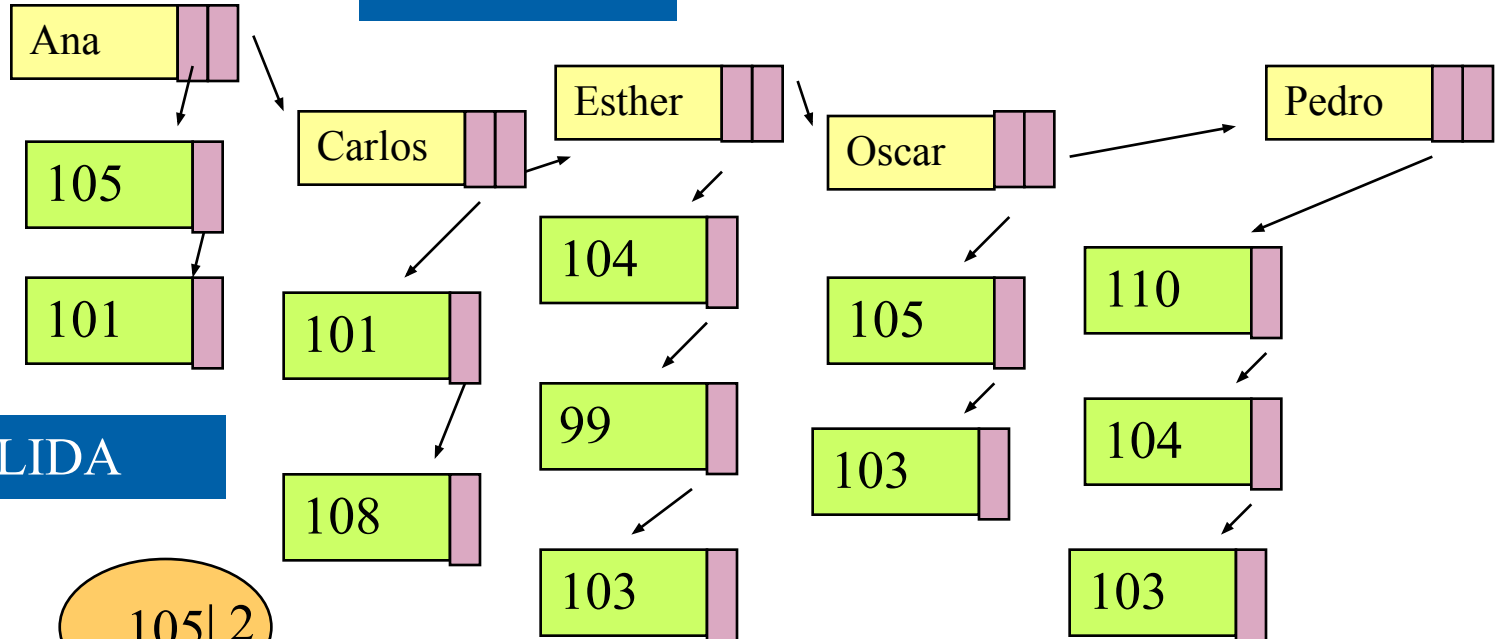
Para cada persona se tiene una lista de los cursos ya aprobados (los cursos aparecen identificados por su código).

Se pide generar un árbol binario ordenado de cursos donde para cada curso se tenga la cantidad de alumnos que lo aprobó.

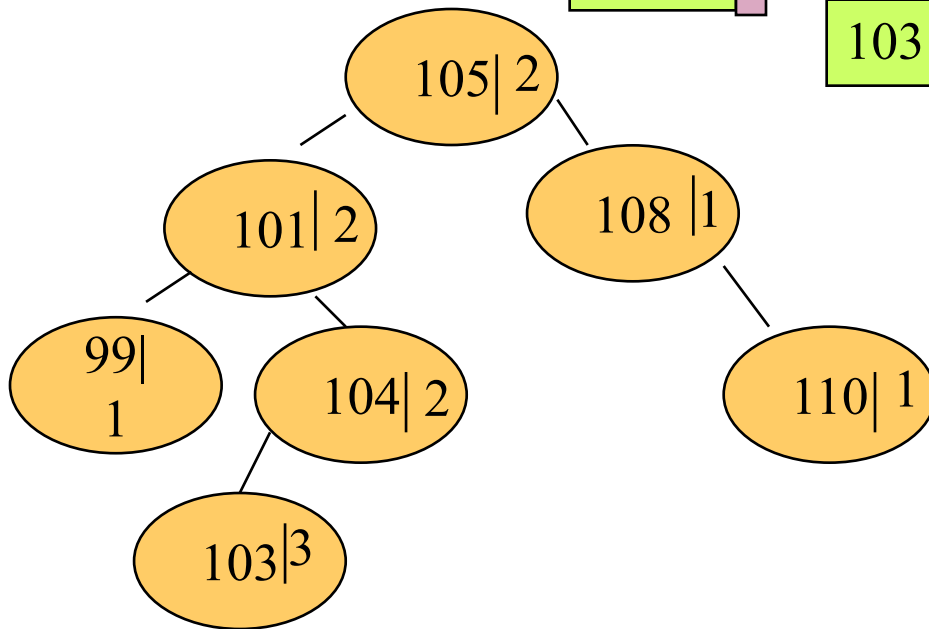
Nota: El orden del árbol es por código de curso.

# Ejercitación Árboles

ENTRADA



SALIDA



¿Qué pasa si el alumno no tiene ninguna materia aprobada?



# Ejercitación Árboles

---

## Una solución posible...

*{Recorrer la lista de alumnos}*

*{Recorrer la lista de cursos del alumno}*

*{Buscar el curso en el árbol binario}*

*{Si está entonces aumentar en 1 el total de alumnos del curso}*

*{Sino Insertar en el árbol binario ordenado}*

# Ejercitación Árboles

## Type

ListaCursos = ^nodocurso;

Nodocurso = **record**

    Codigo:string [4];

    Sig : ListaCursos;

**End;**

ListaAlumnos = ^Alumno;

Alumno = **record**

    Apellido: string [20];

    Cursos: ListaCursos;

    Sig : ListaAlumnos;

**End;**

ArbolCursos = ^NodoArbol;

NodoArbol = **record**

    Codigo : string[4];

    Cant : integer;

    Izq : ArbolCursos;

    Der: ArbolCursos;

**End;**

## Var

LC, listacur: ListaCursos;

LA, listaalu: ListaAlumnos;

AC, arbol : ArbolCursos;



# Ejercitación Árboles

## **Begin**

*{crear la lista de alumnos con los cursos aprobados}*

*{Recorrer la lista de alumnos}*

Listaalu := LA;

AC := Nil;

**While** (Listaalu <> Nil) **do begin**

*{Recorrer la lista de cursos del alumno}*

Listacur := Listaalu^.cursos;

**While** (Listacur<> Nil) **do begin**

*{buscar el curso en el árbol binario}*

arbol := Buscar (AC, listacur^.codigo);

*{Si está entonces aumentar en 1 el total de alumnos del curso}*

**If** arbol <> nil **then** arbol^.cant := arbol^.cant + 1

*{Sino Insertar en el árbol binario ordenado}*

**Else** Insertar (AC, listacur^.codigo);

Listacur := listacur^.sig;

**End;**

Listaalu := listaalu^.sig;

**End;**

**End.**



# Árboles Binarios Ordenados - Operaciones

- Operación Buscar: localiza un elemento X en el árbol

**Function Buscar (A: ArbolCursos; x:integer) : ArbolCursos**

Esta operación retorna un puntero al nodo en el árbol A que tiene valor "x" o Nil si no existe.

- Operación Insertar:

**Procedure Insertar (Var A : ArbolCursos; dato : integer);**

Inserta el código en el árbol A. Es igual al insertar de la teoría solo que tendríamos que poner el campo cant en 1 (uno) dentro del código del caso base (cuando llegamos a A = nil).

**Ver su implementación en la teoría**

# Ejercitación Árboles

- a) Informar la cantidad de alumnos que tienen los cursos con código menor a 105.

```
Procedure cantidades (a: arbolcursos);  
begin  
  If a <> Nil then  
    If (a^.codigo < 105) then begin  
      write (a^.cant);  
      Cantidades (a^.der);  
      Cantidades (a^.izq);  
    End  
    Else cantidades (a^.izq);  
End;
```

*¿Funciona para todos los casos?*



# Ejercitación Árboles

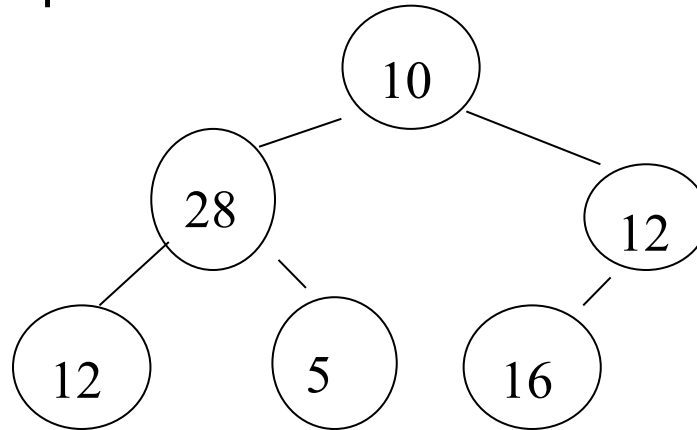
- b) Informar cuales son las cantidades de alumnos de los cursos con código comprendidos entre 105 y 110.

```
Procedure intervaloCantidades (a: arbolcursos);  
begin  
  If a <> Nil then  
    if (a^.codigo >= 105) then  
      if (a^.codigo <= 110) then begin  
        Write (a^.cant);  
        intervaloCantidades (a^.izq);  
        intervaloCantidades (a^.der);  
      End  
      Else intervaloCantidades (a^.izq);  
    Else intervaloCantidades (a^.der);  
End;
```

*¿Funciona para todos los casos?*

# Ejercitación Árboles

2. Dado un árbol binario (OBSERVAR QUE NO ESTÁ ORDENADO) donde cada uno de sus nodos tiene como dato un número entero. Ejemplo: 10 28 12 5 16 12



- a) Imprimir el peso de cada rama del árbol en términos de la suma de los valores que contienen sus nodos.

# Ejercitación Árboles

## type

```
Arbol = ^nodo;  
Nodo = record  
    Dato: integer;  
    Izq: arbol;  
    Der: arbol;  
End;
```

## Var

```
A: arbol; cant: integer;  
max : integer {para parte (b)}
```

**Procedure** Imprimir (a: arbol; cant: integer);

## Begin

```
Cant := cant + a^.dato;  
If (a^.izq <> Nil) then Imprimir (a^.izq, cant);  
If (a^.der <> Nil) then Imprimir (a^.der, cant);  
If (a^.izq = Nil) and (a^.der = Nil) then  
    write ("La suma de la rama es:", cant);
```

**End;**

{prog. ppal}

## Begin

....

Cant := 0;

**If** (a <> nil) **then**  
 Imprimir (a, cant);

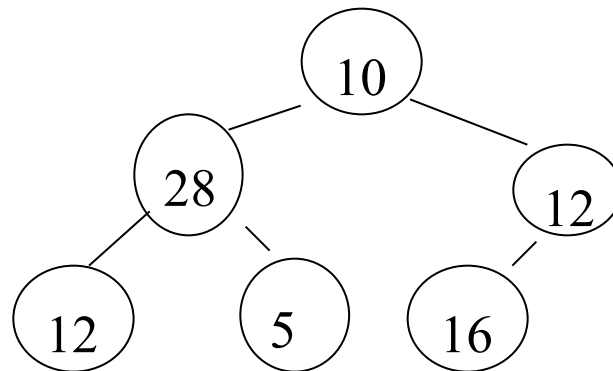
...

**End.**

# Ejercitación Árboles

2.b) Supongamos que se desea calcular la rama del árbol que tiene más peso en términos de la suma de los valores que contienen sus nodos. Se pide informar la suma máxima.

Resultado: La rama que tiene máxima suma es la que totaliza 50





# Ejercitación Árboles

```
Procedure ImprimirModificado (a: arbol; cant: integer;  
                             var max:integer);
```

```
Begin
```

```
  Cant := cant + a^.dato;
```

```
  If (a^.izq <> Nil) then ImprimirModificado (a^.izq, cant, max);
```

```
  If (a^.der <> Nil) then ImprimirModificado (a^.der, cant, max);
```

```
  If (a^.izq = Nil) and (a^.der = Nil) then
```

```
    If cant > max then max := cant;
```

```
End;
```

```
{Prog Ppal}
```

```
Begin
```

```
  ....
```

```
  max:= -1; Cant := 0;
```

```
  if (a <> nil) then ImprimirModificado (a, cant, max);
```

```
  Write (max);
```

```
  ...
```

```
End.
```