Técnico Informático

ESTRUCTURA DE DATOS

UNIDAD VI: INTRODUCCIÓN A ÁRBOLES

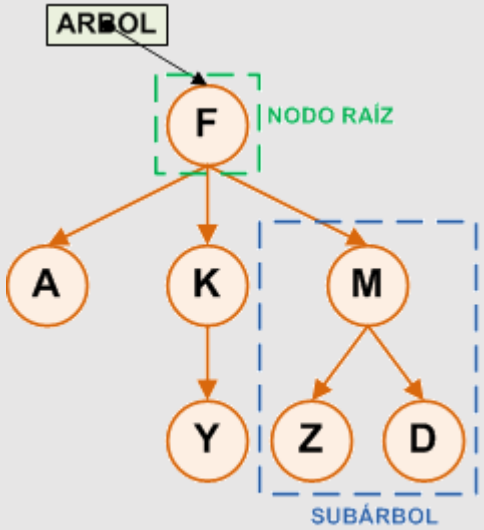
Escuela de Minas "Dr. Horacio Carrillo"
Universidad Nacional de Jujuy

Índice

- Conceptos básicos
- Árboles Binarios
- Árboles Binarios de Búsqueda
- Operaciones: recorrido, búsqueda, inserción y eliminación
- Aplicaciones

Conceptos Básicos (1)

- Un TDA árbol es un conjunto finito de nodos, dónde cada nodo está formado por *n* enlaces (árbol de grado *n*), tal que:
  1. existe un nodo especial llamado ***RAIZ***
  2. existen ***n>0*** subconjunto de nodos llamados **subárboles.**
- Un árbol se dice **vacío** si no tiene nodos.



Árbol de grado 3

Conceptos Básicos (2)

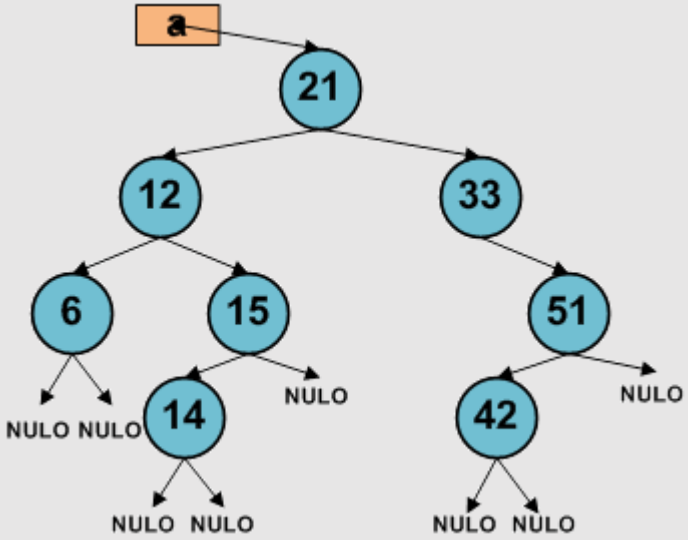
- **Nodos o vértices**: se trata de registros compuestos por un campo de datos y *n* punteros o enlaces.
- **Aristas o arcos**: enlaces que conectan los nodos.
- **Raíz**: es el primer nodo del árbol, a partir de éste descienden todos los demás.
- **Nodos terminales u hojas**: son los nodos que no tienen descendientes.
- **Nodos interiores**: son los nodos que tienen al menos un subárbol.

Conceptos Básicos (3)

- **Ascendiente**: un nodo *x* es ascendiente (padre) de un nodo *y*, si *x* tiene un puntero al nodo *y*.
- **Descendiente**: un nodo *b* es descendiente (hijo) de un nodo *a*, si *b* es apuntado por el nodo *a*.
- **Camino**: conjunto de enlaces entre 2 nodos.
- **Rama**: camino existente entre la raíz y un nodo hoja.
- **Nivel de un nodo**: cantidad de aristas entre la raíz y un nodo específico.
- **Altura o profundidad**: cantidad de nodos de la rama más larga (máximo nivel más 1).

Conceptos Básicos (4)

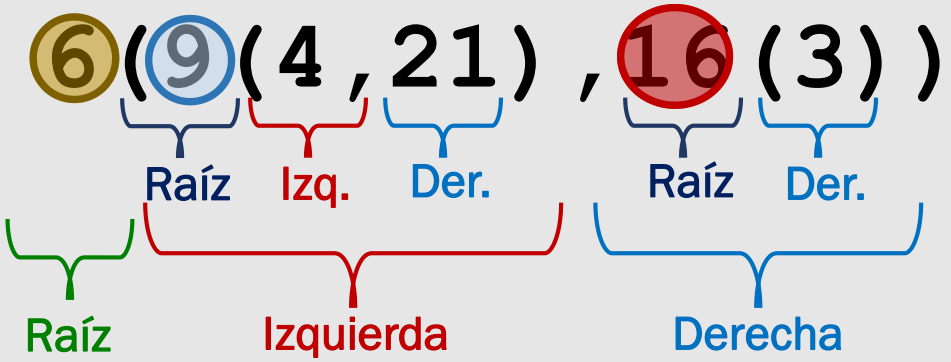
- **Peso del árbol**: cantidad de nodos **hojas** del árbol.
- **Grado de un árbol**: máximo N° de **descendientes** de un nodo.
- **Árbol similar**: árboles con la **misma estructura**.
- **Árbol equivalente**: árboles con la **misma estructura y datos**.

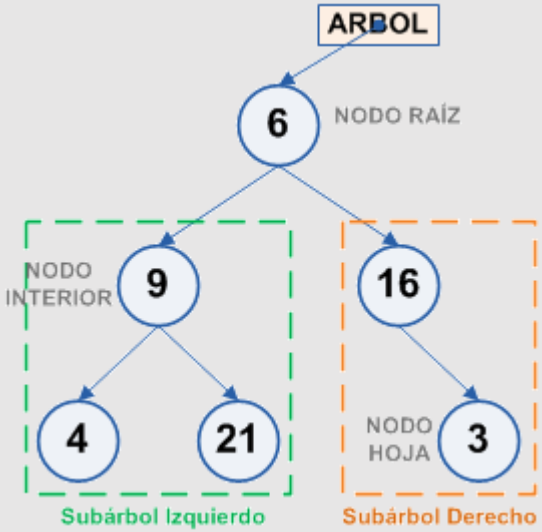



```
Raíz: 21
Hojas: 6,14,42
Nodos interiores: 12,15,33,51
Nivel de nodo 51: 2
Altura: 4
Peso: 3
Grado: 2
```

Árboles Binarios (1)

- Un árbol binario es un árbol de grado 2 tal que:
  1. existe un nodo llamado nodo RAÍZ
  2. cada nodo del árbol tiene como máximo 2 descendientes (subárbol izquierdo y subárbol derecho)
- Representación: paréntesis anidados y gráfica





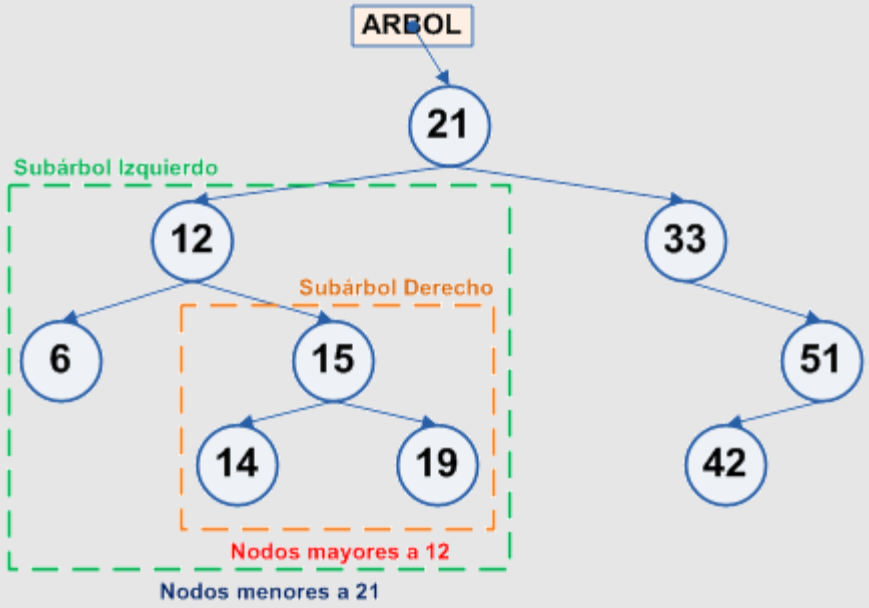
Árboles Binarios (2)

- Implementación

```
typedef struct tnodo *pnodo;
typedef struct tnodo
                        {
                         tipo_dato dato;
                         pnodo izquierdo;
                         pnodo derecho;
                        };
```

Árboles Binarios de Búsqueda

- Un árbol binario de búsqueda es un árbol binario en el que dadas dos condiciones mutuamente excluyentes, para cada nodo, todas la claves de su subárbol izquierdo satisfacen una condición y todas las de su subárbol derecho la otra.

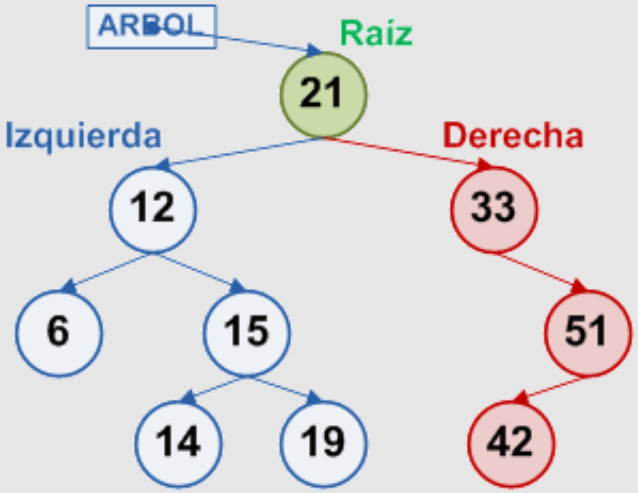


Operaciones (1)

- Recorrido
  - Pre-orden (RAÍZ-izquierda-derecha)
  - En-orden (izquierda-RAÍZ-derecha)
  - Pos-orden (izquierda-derecha-RAÍZ)
- Búsqueda (árboles binarios de búsqueda)
- Inserción (árboles binarios de búsqueda)
- Eliminación (árboles binarios de búsqueda)

Operaciones (2)

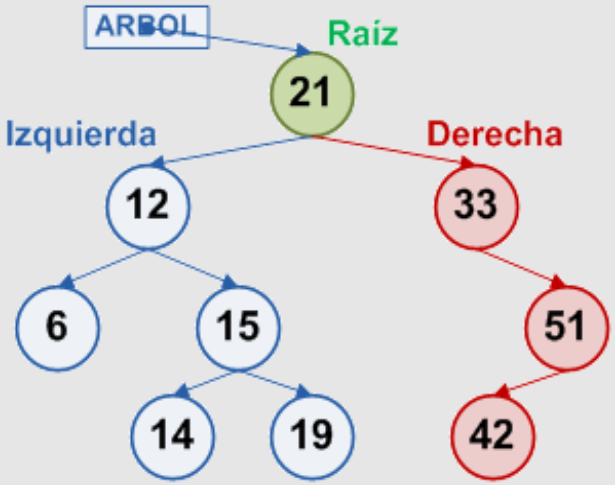
- Recorrido
  - Pre-orden (RAÍZ-izquierda-derecha)
  - En-orden (izquierda-RAÍZ-derecha)
  - Pos-orden (izquierda-derecha-RAÍZ)



Operaciones (3)

- Recorrido
  - Pre-orden (RAÍZ-izquierda-derecha)

```
void preorden(pnodo a)
{ if (a!=NULL)
    { cout << a->dato;
      preorden(a->izq);
      preorden(a->der);
    }
}
```

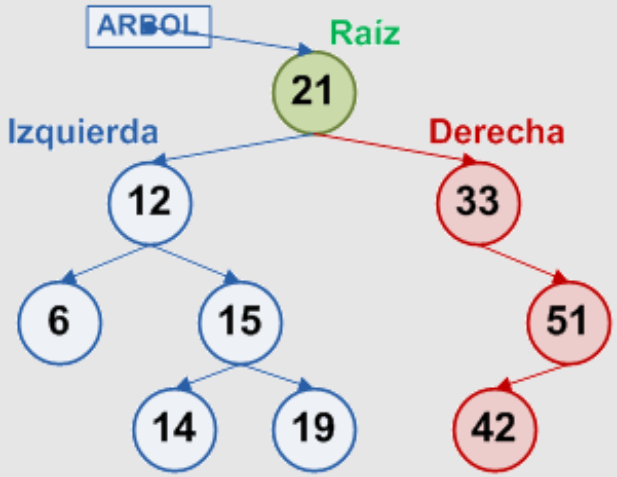



```
Pre-orden:21,12,6,15,14,19,33,51,42
```

Operaciones (4)

- Recorrido
  - En-orden (izquierda-RAÍZ-derecha)

```
void enorden(pnodo a)
{ if (a!=NULL)
    { enorden(a->izq);
      cout << a->dato;
      enorden(a->der);
    }
}
```

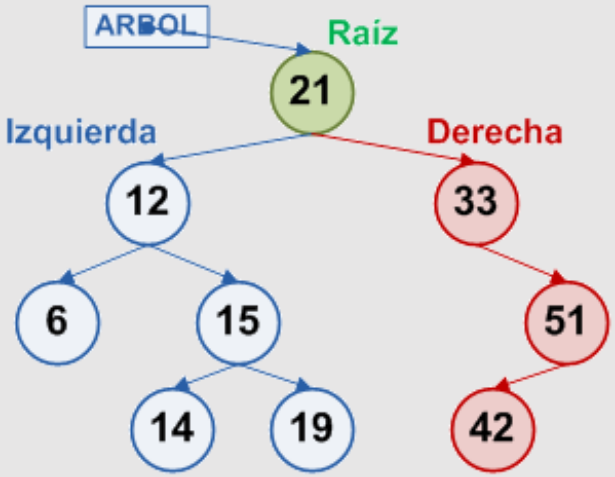



```
En-orden: 6,12,14,15,19,21,33,42,51
```

Operaciones (5)

- Recorrido
  - Pos-orden (izquierda-derecha-RAÍZ)

```
void posorden(pnodo a)
{ if (a!=NULL)
    { posorden(a->izq);
      posorden(a->der);
      cout << a->dato;
    }
}
```




```
Pos-orden: 6,14,19,15,12,42,51,33,21
```

Operaciones (6)

- Búsqueda en árboles binarios

```
bool busqueda(pnodo a,int buscado)
{ bool encontrado=false;
  if (a!=NULL)
  { if (a->dato==buscado)
     encontrado=true;
    else
     { encontrado=busqueda(a->izq,buscado);
       if (encontrado==false)
         encontrado=busqueda(a->der,buscado);}
  }
  return encontrado;
}
```

Operaciones (7)

- Búsqueda en árboles binarios de **búsqueda**

```
bool busqueda(pnodo a,int buscado)
{ bool encontrado=false;
  if (a!=NULL)
  { if (a->dato == buscado)
     encontrado=true;
    else
      if (buscado < a->dato)
         encontrado=busqueda(a->izq,buscado);
      else
         encontrado=busqueda(a->der,buscado);
  }
  return encontrado;
}
```

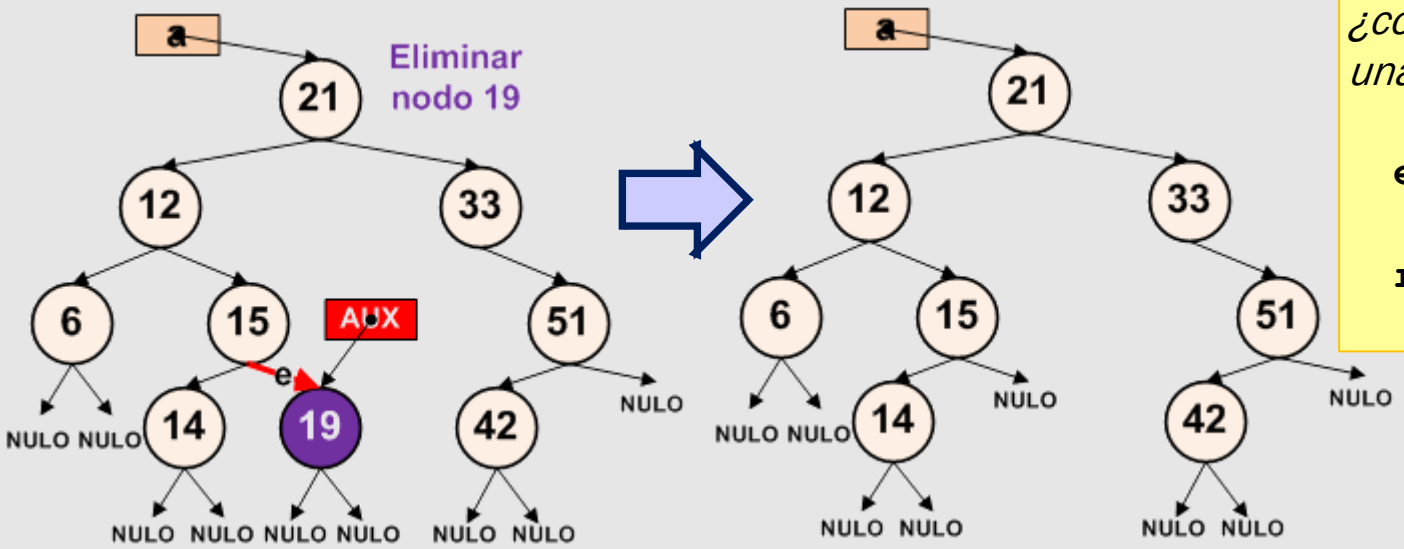
Operaciones (8)

- Inserción en árboles binarios de búsqueda

```
void insercion(pnodo &a,pnodo nuevo)
{
  if (a==NULL)
    a=nuevo;
  else
    if (nuevo->dato < a->dato)
        insercion(a->izq,nuevo);
      else
        insercion(a->der,nuevo);
}
```

Operaciones (9)

- Eliminación en árboles binarios de búsqueda
  - *Caso 1:* Eliminar un nodo hoja
  - *Caso 2:* Eliminar un nodo con un solo descendiente
  - *Caso 3:* Eliminar un nodo con 2 descendientes

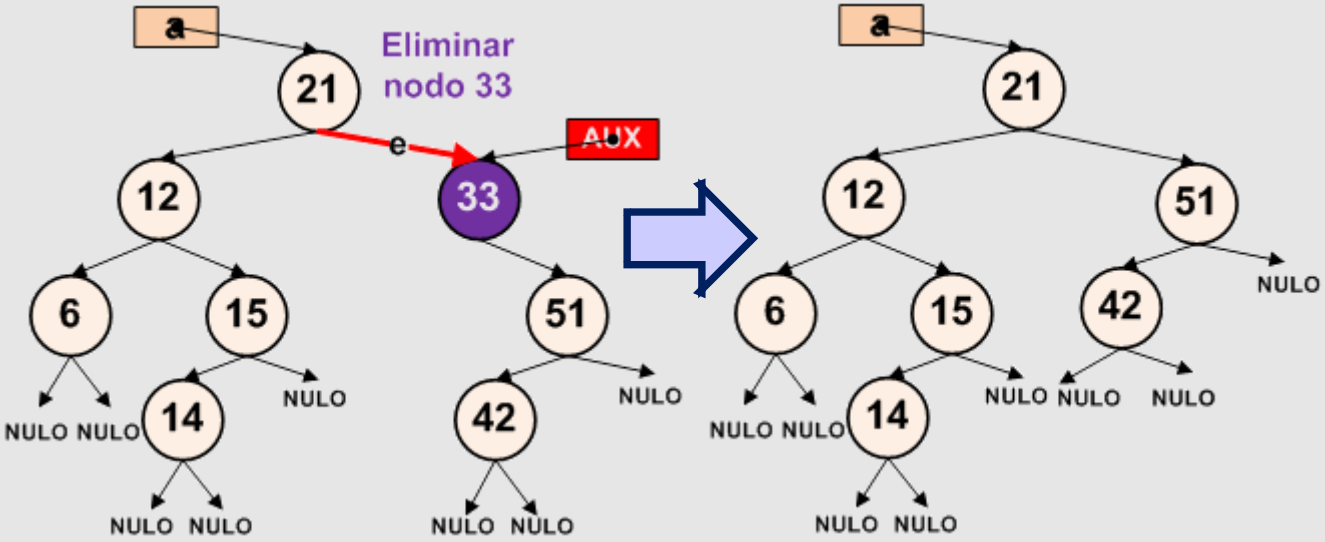


¿cómo se elimina una hoja?

```
aux=e;
e=aux->izq;
...
return aux;
```

Operaciones (10)

- Eliminación en árboles binarios de búsqueda
  - *Caso 1:* Eliminar un nodo hoja
  - *Caso 2:* Eliminar un nodo con un solo descendiente
  - *Caso 3:* Eliminar un nodo con 2 descendientes

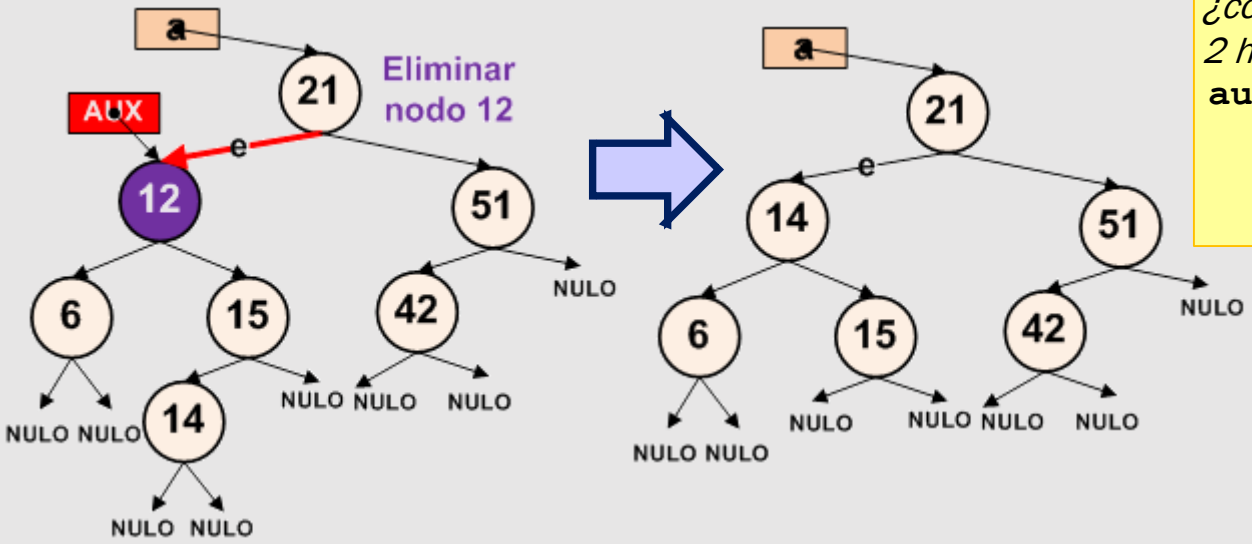


¿cómo se elimina un nodo con 1 hijo?

```
aux=e;
e=aux->der;
...
return aux;
```

Operaciones (11)

- Eliminación en árboles binarios de búsqueda
  - *Caso 1:* Eliminar un nodo hoja
  - *Caso 2:* Eliminar un nodo con un solo descendiente
  - *Caso 3:* Eliminar un nodo con 2 descendientes



¿cómo se elimina un nodo con 2 hijos?

```
aux=men_may(e,e->der);
...
return aux;
```

Operaciones (12)

- Eliminación en árboles binarios de búsqueda

```
pnodo eliminar(pnodo &a,int valor)
{pnodo aux;
 if (a==NULL)
  aux=NULL;
 else
  if (a->dato > valor)
   aux=eliminar(a->izq,valor);
  else
   if (a->dato < valor)
    aux=eliminar(a->der,valor);
   else
{ aux=a;
  if (a->izq==NULL)
   a=a->der;
  else
   if (a->der==NULL)
    a=a->izq;
   else
    aux=menor_mayores(a,a->der);
}
 return aux;
}
```

Operaciones (13)

- Eliminación en árboles binarios de búsqueda

```
pnodo menor_mayores(pnodo elegido,pnodo &menor)
{ pnodo aux;
  if (menor->izq!=NULL)
    aux=menor_mayores(elegido,menor->izq);
  else
    { cambio(elegido->dato,menor->dato);
      aux=menor;
      menor=menor->der;
    }
  return aux;
}
```

Se recorre el subárbol derecho buscando el menor valor.

Se sustituye el valor a eliminar por el menor de los mayores y se actualizan punteros.

Aplicaciones

- Bases de datos
- Indexación de archivos
- Sistemas de archivos
- Sistemas operativos (jerarquías de procesos)
- Conversión de expresiones
- Clasificación de información

Bibliografía

- Joyanes Aguilar *et al*. Estructuras de Datos en C++. Mc Graw Hill. 2007.
- De Giusti, Armando *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta. 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Hernández, Roberto *et al*. Estructuras de Datos y Algoritmos. Prentice Hall. 2001.