Programación UNSAM
Estructuras de datos dinámicas

David López

v. 2020

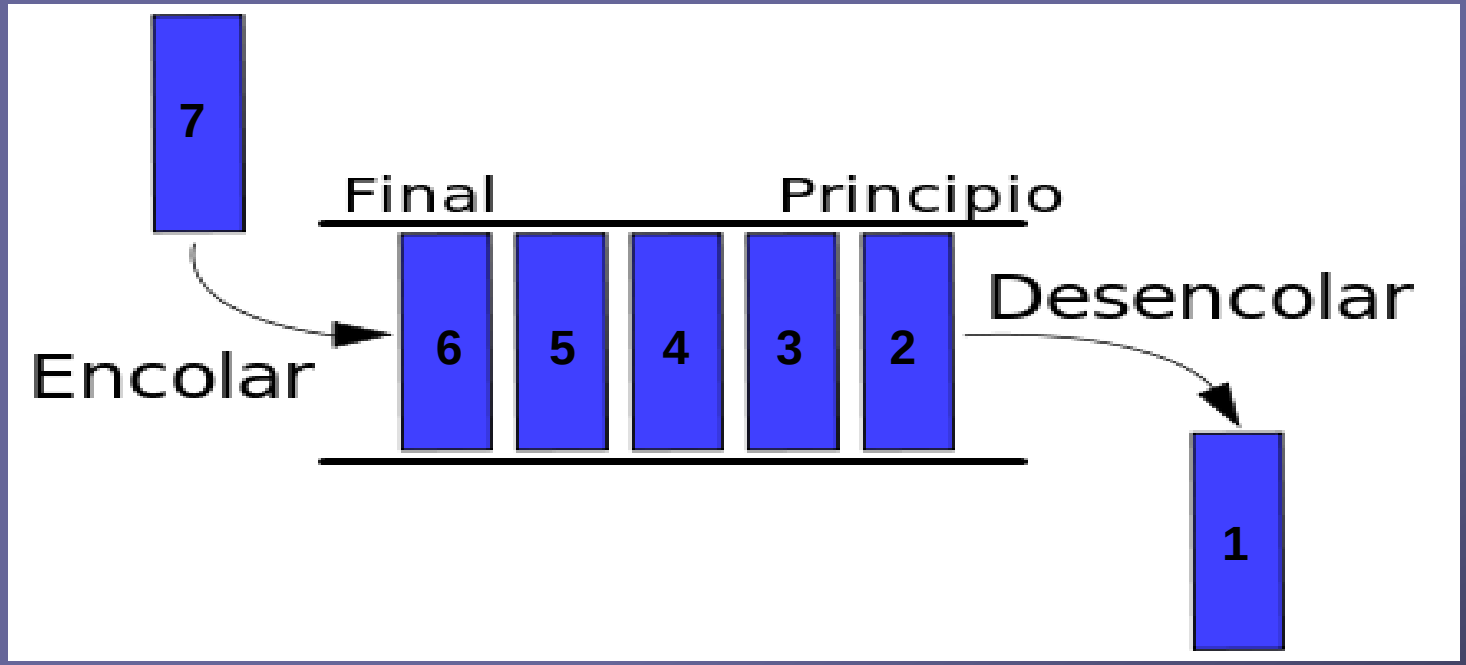
Estructuras de datos dinámicas

- Estructura de datos:
  - Forma de organizar datos en una computadora para su uso de manera eficiente
  - Ejemplos: vector, matriz, estructura (struct), unión
- Estructura de datos dinámica
  - Es flexible: su tamaño puede aumentar o decrecer según la necesidad
  - Ejemplos: cola, pila, lista enlazada, árbol

Cola (queue)

- Solamente permite dos operaciones
  - Encolar (enqueue) : poner un dato
  - Desencolar (dequeue): sacar un dato
- Los datos solamente se pueden acceder (desencolar) en el orden en que fueron almacenados (encolados)
  - Propiedad FIFO (First In First Out)

Cola



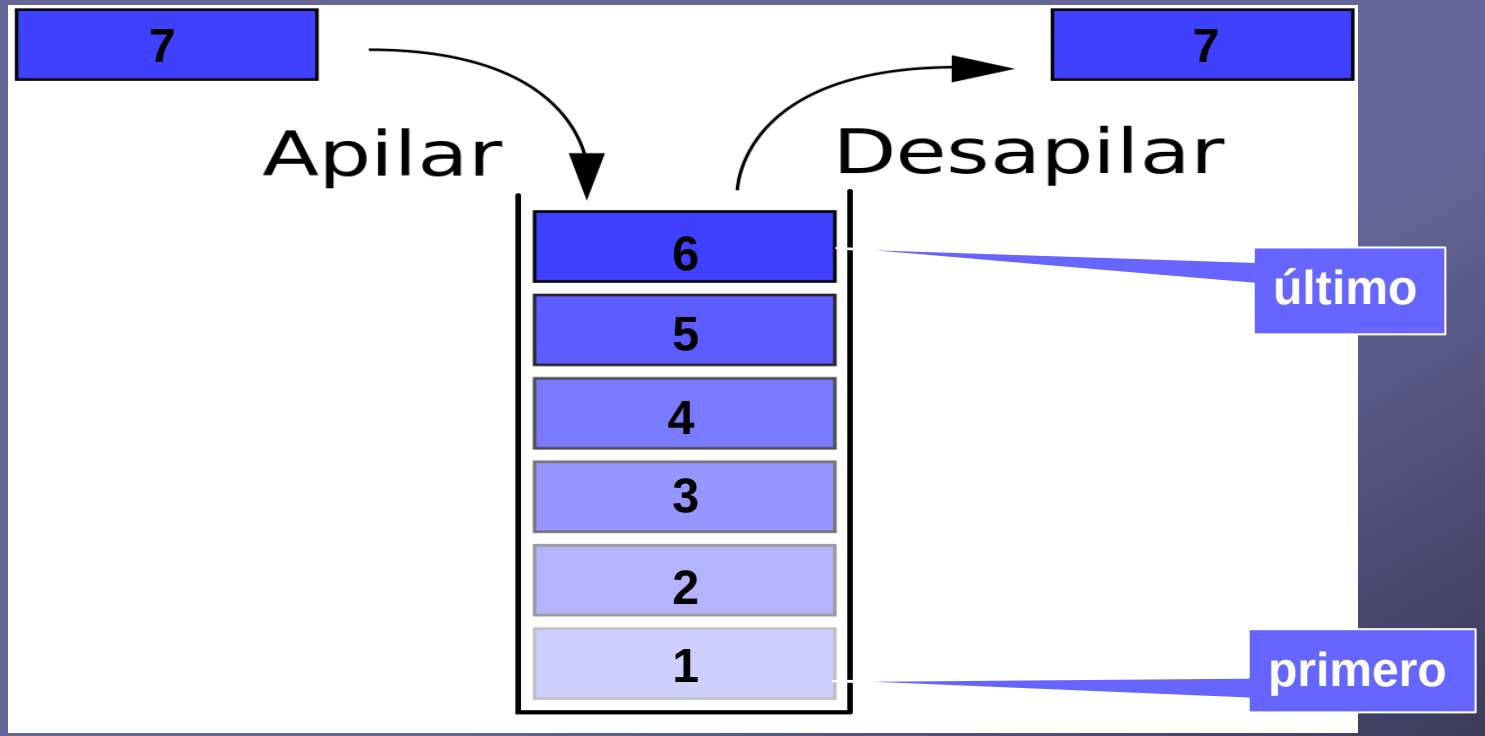
Cola

- Ejemplo de uso: cola de impresión
  - En una impresora los trabajos se imprimen en el orden en que fueron enviados

Pila (stack)

- Solamente permite dos operaciones
  - Apilar (push) : poner un dato
  - Desapilar (pop): sacar un dato
- Los datos solamente se puede acceder (desapilar) en el orden inverso en que fueron almacenados (apilados)
  - Propiedad LIFO (Last In First Out)

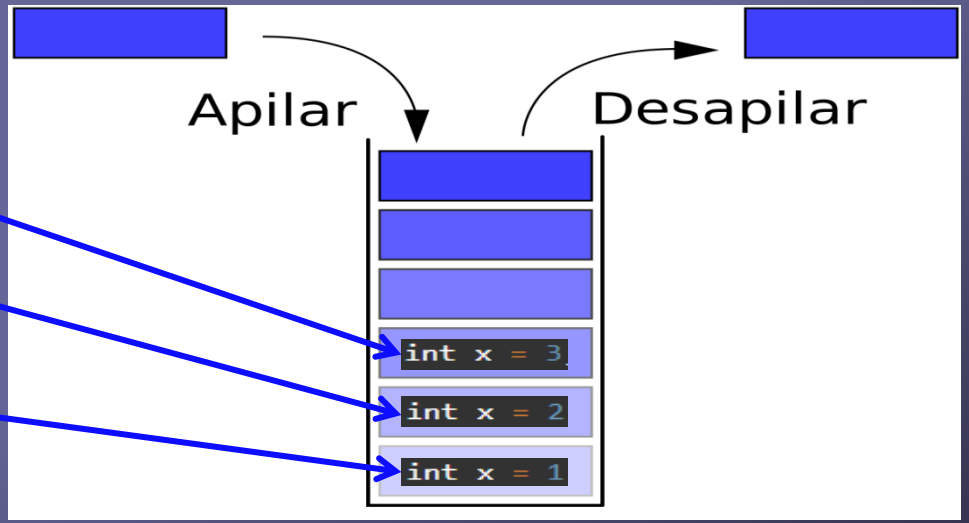
Pila



Pila

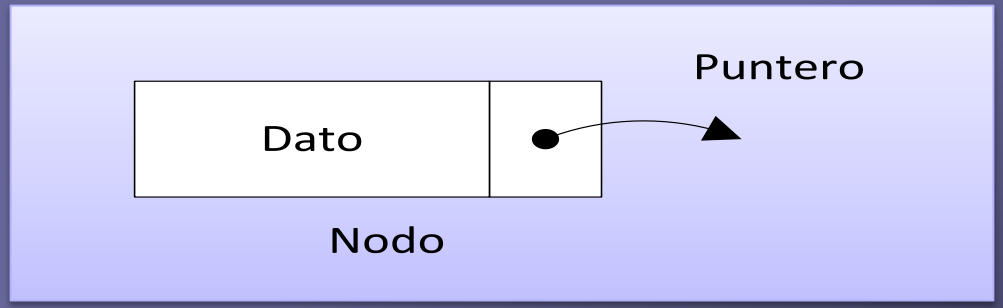
- Ejemplo de uso: memoria de un programa en C
  - En la zona de memoria stack se van apilando las variables y parámetros de las funciones invocadas

```
void func2() {
    int x = 3;
}

void func1() {
    int x = 2;

    func2();
}

int main() {
    int x = 1;

    func1();
    return 0;
}
```



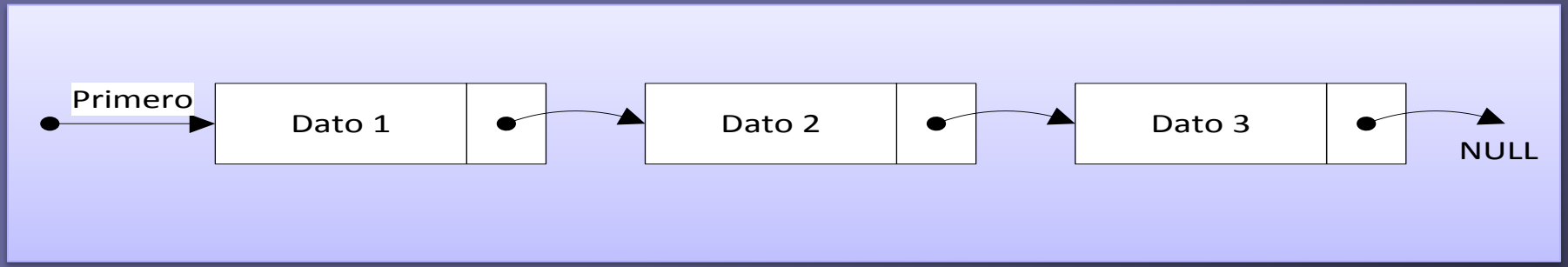
Lista enlazada (linked list)

- Está formada por nodos conectados
- Los nodos está compuestos por dos partes
  - Datos
  - Puntero que enlaza con otro nodo



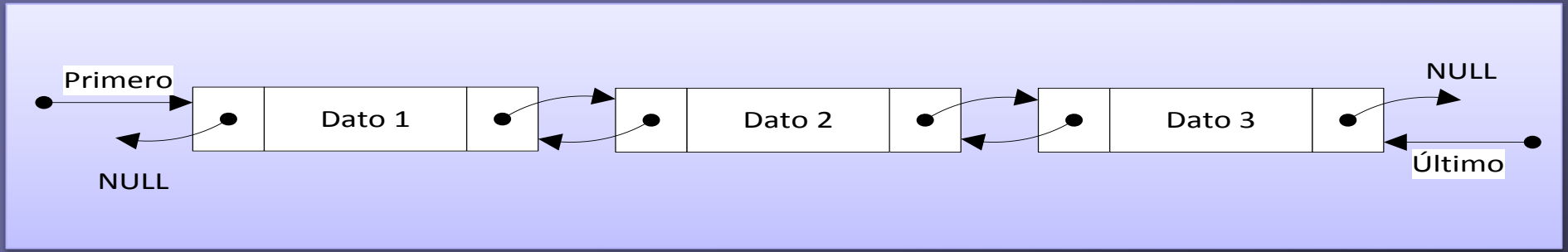
Lista simplemente enlazada

- Tiene un puntero llamado primero
- Se empieza a recorrer por el principio
- Cada nodo se conecta con el siguiente mediante un puntero



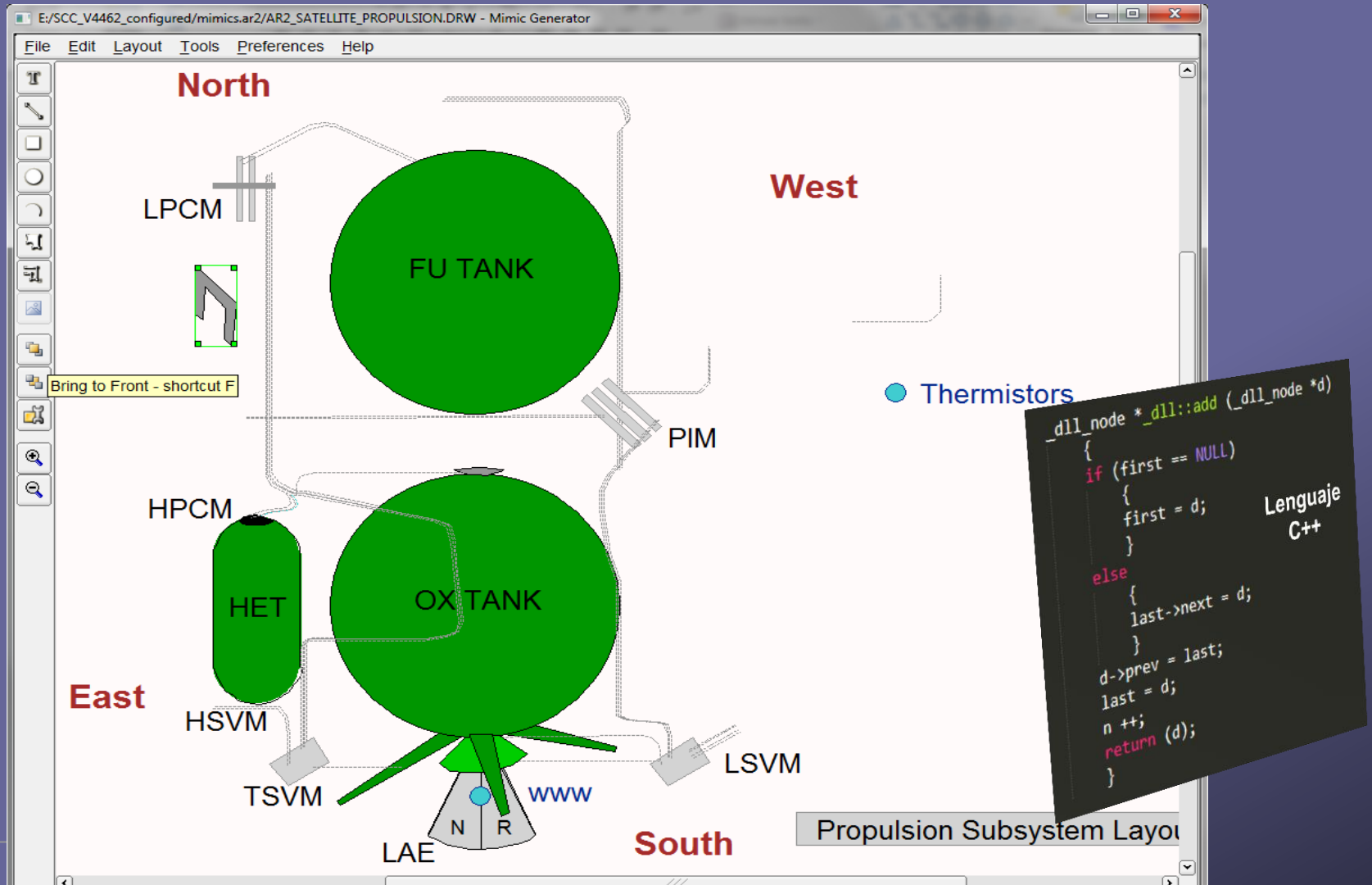
Lista doblemente enlazada

- Tiene 2 punteros: primero y ultimo
- Se puede empezar a recorrer ambos extremos
- Cada nodo se conecta con el siguiente y con el anterior mediante 2 punteros



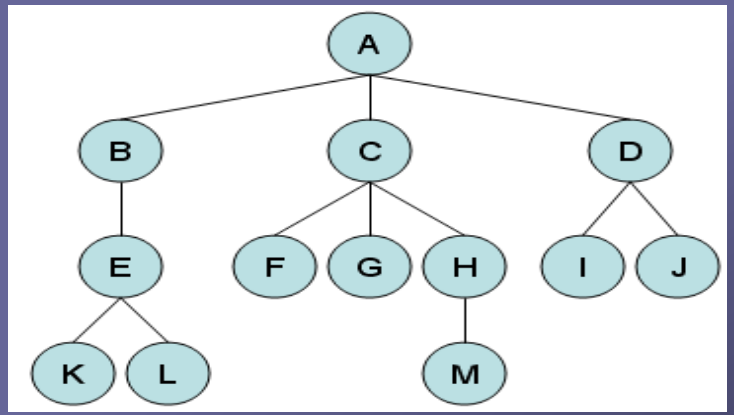
Aplicaciones de las listas

- Examen parcial y final de esta materia 😜
- Casos donde se necesite almacenar una cantidad variable de elementos
- Especialmente si va a haber inserciones y/o eliminaciones frecuentes
  - Por ejemplo un editor de objetos gráficos como el de la diapositiva siguiente (generador de pantallas del sistema de control satelital de Arsat)


E:/SCC_V4462_configured/mimics.ar2/AR2_SATELLITE_PROPULSION.DRW - Mimic Generator
File Edit Layout Tools Preferences Help
North
West
East
South
LPCM
FU TANK
PIM
HPCM
HET
OX TANK
HSVM
TSVM
LSVM
LAE
N
R
www
Thermistors
Bring to Front - shortcut F
Propulsion Subsystem Layou
_dll_node *_dll::add (_dll_node *d)
{
if (first == NULL)
{
first = d;
}
else
{
last->next = d;
}
d->prev = last;
last = d;
n ++;
return (d);
}
Lenguaje
C++

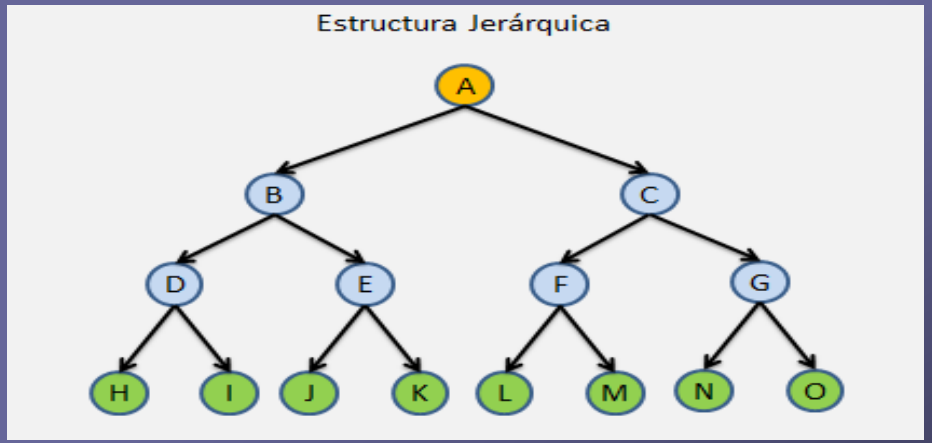
Árbol (tree)

- Matemáticamente es un grafo direccional sin ciclos
- Se compone de nodos con datos, de los que pueden depender otros nodos hijos



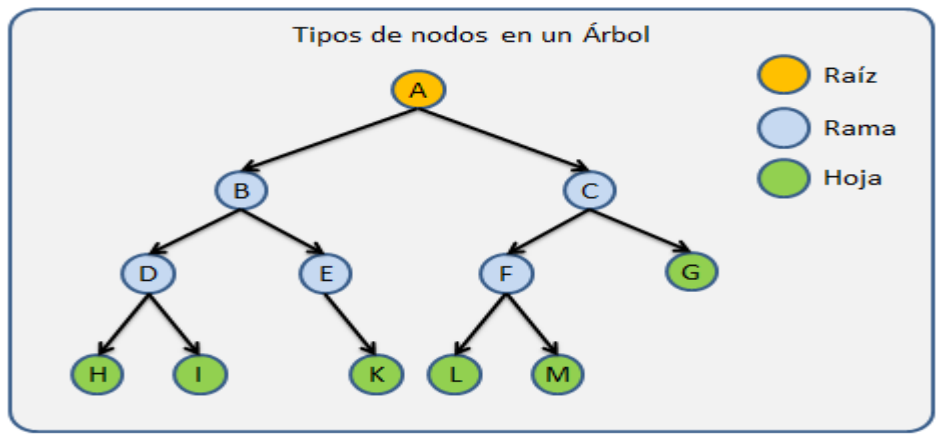
Árbol

- Tiene una estructura jerárquica ramificada
- Se empieza a recorrer por un nodo llamado raíz
- Cada nodo está conectado mediante punteros con uno o más nodos hijos



Árbol

- Un caso particular es el árbol binario, donde cada nodo sólo puede tener 2 hijos como máximo

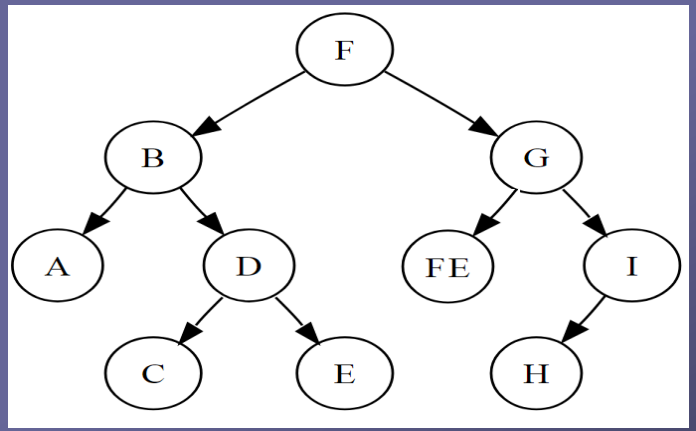


Árbol

- Aplicaciones
  - Representación de jerarquías
    - Ejemplo: organigrama
  - Resolución de expresiones en compiladores
    - Ejemplo: x = 2 * 3 + 8 / 2
  - Ordenamiento
  - Índices de bases de datos (ej. MySQL)

Árbol: ordenamiento y búsqueda

- El siguiente árbol está ordenado (de izquierda a derecha, recorrido *inorden*)
- ¿Se puede insertar el elemento "FE" manteniendo el orden?



FE