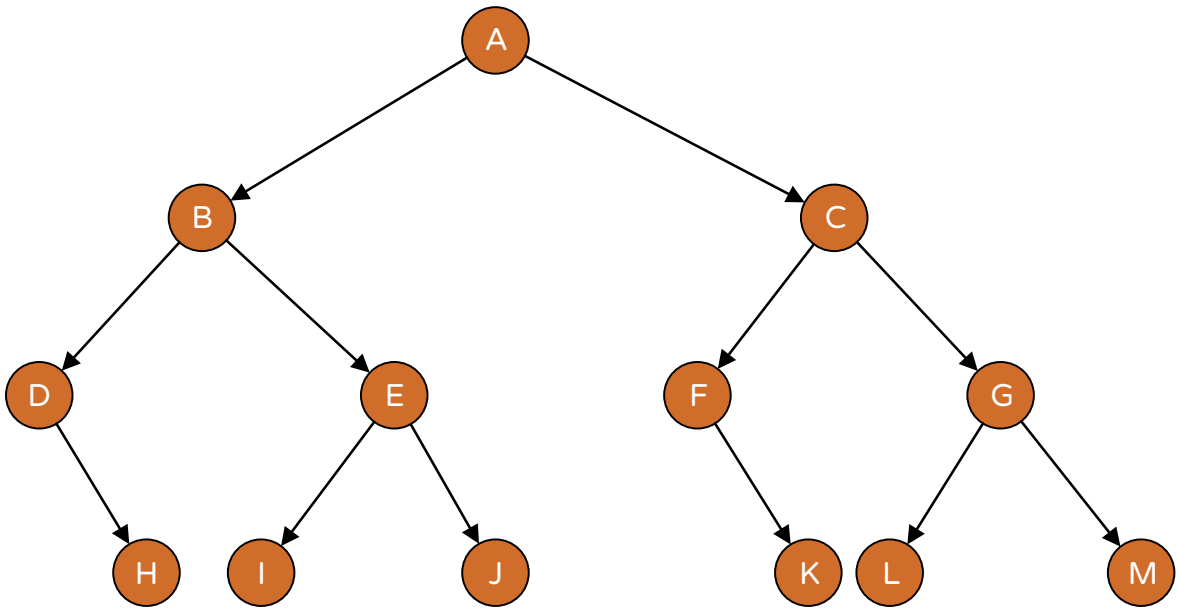
Cómo guardar un árbol en un archivo.

Se puede idear diversas estrategias para guardar un árbol en un archivo. La que emplearemos consiste en recorrer exhaustivamente el árbol y por cada nodo se almacena el dato, quién es el padre (identificador del padre) y si el nodo a ser guardado está enlazado por la derecha o izquierda con el padre. En un principio, como el nodo raíz no tiene padre y por lo tanto no tiene con quien enlazarse, se empleará el asterisco como indicador de este hecho.

Supongamos el siguiente árbol:



Para recorrerlo exhaustivamente, se puede emplear uno de 3 algoritmos de recorrido: PreOrden, InOrden y PostOrden. Los tres se encontrarán implementado en módulos recursivos.

```
01  # -*- coding: utf-8 -*-
02  ###############################################################################
03  # DEFINICIONES:
04  class TipoNodo:
05        Dato = ""
06        Izq  = None
07        Der  = None
08
09  ###############################################################################
10  # MÓDULOS:
11
12  def CrearArbol():
13      Raiz       = TipoNodo()
14      Raiz.Dato  = "A"
15
16      NodoB      = TipoNodo()
17      NodoB.Dato = "B"
18      Raiz.Izq   = NodoB
19
20      NodoC      = TipoNodo()
21      NodoC.Dato = "C"
22      Raiz.Der   = NodoC
23
24      NodoD      = TipoNodo()
25      NodoD.Dato = "D"
```

```
26      NodoB.Izq   = NodoD
27
28      NodoE      = TipoNodo()
29      NodoE.Dato = "E"
30      NodoB.Der  = NodoE
31
32      NodoF      = TipoNodo()
33      NodoF.Dato = "F"
34      NodoC.Izq  = NodoF
35
36      NodoG      = TipoNodo()
37      NodoG.Dato = "G"
38      NodoC.Der  = NodoG
39
40      NodoH      = TipoNodo()
41      NodoH.Dato = "H"
42      NodoD.Der  = NodoH
43
44      NodoI      = TipoNodo()
45      NodoI.Dato = "I"
46      NodoE.Izq   = NodoI
47
48      NodoJ      = TipoNodo()
49      NodoJ.Dato = "J"
50      NodoE.Der  = NodoJ
51
52      NodoK      = TipoNodo()
53      NodoK.Dato = "K"
54      NodoF.Der  = NodoK
55
56      NodoL      = TipoNodo()
57      NodoL.Dato = "L"
58      NodoG.Izq   = NodoL
59
60      NodoM      = TipoNodo()
61      NodoM.Dato = "M"
62      NodoL.Der  = NodoM
63      return Raiz
64
65  def PreOrden(Punt, Archivo, Padre, Direccion):
66      if Punt != None:
67         Archivo.write(Punt.Dato + ";" + Padre + ";" + Direccion + "\n")
68         PreOrden(Punt.Izq, Archivo, Punt.Dato, "I")
69         PreOrden(Punt.Der, Archivo, Punt.Dato, "D")
70
71  def InOrden(Punt, Archivo, Padre, Direccion):
72      if Punt != None:
73         InOrden(Punt.Izq, Archivo, Punt.Dato, "I")
74         Archivo.write(Punt.Dato + ";" + Padre + ";" + Direccion + "\n")
75         InOrden(Punt.Der, Archivo, Punt.Dato, "D")
76
77  def PostOrden(Punt, Archivo, Padre, Direccion):
78      if Punt != None:
79         PostOrden(Punt.Izq, Archivo, Punt.Dato, "I")
80         PostOrden(Punt.Der, Archivo, Punt.Dato, "D")
81         Archivo.write(Punt.Dato + ";" + Padre + ";" + Direccion + "\n")
82
83
84  ###########################################################################
85  # PROGRAMA PRINCIPAL:
86
87  Raiz = CrearArbol()
88
89  # Guardar empleando PreOrden
90  Pre  = open("PreOrden.txt","w")
```

```
 91  print("\nPreOrden:")
 92  PreOrden(Raiz, Pre, "*", "*")
 93  Pre.close()
 94
 95  # Guardar empleando IOrden
 96  In   = open("InOrden.txt","w")
 97  print("\nInOrden:")
 98  InOrden(Raiz, In, "*", "*")
 99  In.close()
100
101  # Guardar empleando PostOrden
102  Post = open("PostOrden.txt","w")
103  print("\nPostOrden:")
104  PostOrden(Raiz, Post, "*", "*")
105  Post.close()
106
107  input("Fin Programa")
```

Observe que los 3 algoritmos de recorrido son muy similares (líneas 65 a 81), la diferencia está en qué momento se procesa (graba) el nodo.

Los datos quedarían almacenados de la siguiente forma:

PreOrden	**InOrden**	**PostOrden**
A;*;*	D;B;I	H;D;D
B;A;I	H;D;D	D;B;I
D;B;I	B;A;I	I;E;I
H;D;D	I;E;I	J;E;D
E;B;D	E;B;D	E;B;D
I;E;I	J;E;D	B;A;I
J;E;D	A;*;*	K;F;D
C;A;D	F;C;I	F;C;I
F;C;I	K;F;D	M;L;D
K;F;D	C;A;D	L;G;I
G;C;D	L;G;I	G;C;D
L;G;I	M;L;D	C;A;D
M;L;D	G;C;D	A;*;*

Cómo leer un árbol desde un archivo.

De todos los algoritmos de recorrido el más adecuado para guardar el árbol para después poder leerlo es el PreOrden.

El siguiente programa permite realizar la lectura de un archivo con los datos que poblarán el árbol.

```
01  # -*- coding: utf-8 -*-
02  ##########################################################################
03  # DEFINICIONES:
04  class TipoNodo:
05        Dato = ""
06        Izq  = None
07        Der  = None
08
09  ##########################################################################
10  # MÓDULOS:
11  def BuscarPadre(Punt, Padre):
12      global PuntPadre
13      if Punt != None:
14          if Punt.Dato == Padre:
15              PuntPadre = Punt
16          BuscarPadre(Punt.Izq, Padre)
17          BuscarPadre(Punt.Der, Padre)
18
19  def LeerArbol():
20      Archivo   = open("PreOrden.txt","r")
21      Buffer    = Archivo.readline()
22      while Buffer != "":
23          Registro  = Buffer.strip().split(";")
24          Dato      = Registro[0]
25          Padre     = Registro[1]
26          Direccion = Registro[2]
27          if Padre == "*":
28              Raiz      = TipoNodo()
29              Raiz.Dato = Dato
30          else:
31              BuscarPadre(Raiz, Padre)
32              Nodo      = TipoNodo()
33              Nodo.Dato = Dato
34              if Direccion == "I":
35                  PuntPadre.Izq = Nodo
36              else:
37                  PuntPadre.Der = Nodo
38          Buffer = Archivo.readline()
39      return Raiz
40
41  def PreOrden(Punt):
42      if Punt != None:
43          print(Punt.Dato)
44          PreOrden(Punt.Izq)
45          PreOrden(Punt.Der)
46
47  ##########################################################################
48  # PROGRAMA PRINCIPAL:
49  PuntPadre = None
50  Raiz      = LeerArbol()
51  PreOrden(Raiz)
52  input("Fin Programa")
```