void arboles\_manual() {

// se recomienda hacer primero este ejercicio;

puede ser otro o mismo gdb, definir otro typedef nodoT2 para que use ‘dato’ como char

contruir , fig 6.20 a) del libro, construcción 'manual', guia: ver geekforgeek

por esta vez, se puede usar varias flechas como geekforgeek

ej: raiz\_a ->izq-> izq -> der ->dato = 9999;

…

imprimir\_preOrden(raiz\_a);

imprimir\_inOrden(raiz\_a);

imprimir\_posOrden(raiz\_a);

// repetir proceso anterior con fig 6.20 b)

// impresiones deben coincidir con libro;

}

nodoT \*crear\_Arbol( n , m ) {

central = (m-n)/2 +n

if condición paro? return NULL;

? Validar n<=central <= m else return NULL;

nodoT\* nodo= new ...

//central sería la raiz,

nodo->dato = central

Validar central-1 >= n

nodo->izq = crear\_Arbol( n, central-1)

Validar central +1 <=m

nodo->der = crear\_Arbol( central+1 , m )

return nodo;

}

main(){

raiz= crear\_Arbol(10, 40) // se puede ajustar a 50, 40, o 30 para verse en una sóla línea en el preorden;

imprimir\_preOrden2(raiz); // 50 - 25 - 12 - ...

imprimir\_inOrden2(raiz);

imprimir\_posOrden2(raiz);

//crear otro árbol, para el 10!

raiz2= crear\_Arbol( 1, 100) //, puede ajustar a 70, 40, 30, o incluso 20, para verse bien en

// pantalla;

imprimir\_arbol(raiz2, nivel\_i,) // salida con diagonales similar a figura de abajo:

Ejemplo: n=0, m= 100, raiz= 50; [10, 48] 38 nodos, se puede pedir menos nodos

50 2

/ \ 5

25 75 10 terminales , nivel anterior

/ \ / \ / \ / \ max 20 nodos terminales,

12 34 …

...

0

}