

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TAILLE 10

typedef struct noeud NOEUD;

struct noeud
{
    int valeur;
    NOEUD *fils_grand;
    NOEUD *fils_petit;
    NOEUD *parent;
};

void placer_noeud(NOEUD *aplacer, NOEUD *saintpere);
void affiche(NOEUD *saintpere);

int main (void)
{
    NOEUD arbre[TAILLE];
    int i;

    for(i=0; i<TAILLE; i++)
    {
        arbre[i].valeur=TAILLE-i;
        arbre[i].fils_grand=NULL;
        arbre[i].fils_petit=NULL;
        arbre[i].parent=NULL;
    }
    for (i=1; i<TAILLE; i++) placer_noeud(&arbre[i], &arbre[0]);
    // on doit commencer a 1 car le noeud 0 est le pere il ne doit pas etre placer dans l'arbre.

    affiche(arbre);

    system("PAUSE");
    return 0;
}

void placer_noeud(NOEUD *aplacer, NOEUD *saintpere)
{
    // la fonction reçoit un NOEUD en argument et un deuxieme qui représente le noeud père
    NOEUD* pointeur;
    pointeur=saintpere;

    if( aplacer->valeur < saintpere->valeur)
    {

```

```

    if (saintpere->fils_petit == NULL)
    {
        pointeur->fils_petit=aplacer;
        aplacer->parent=pointeur;
    }
    else placer_noeud(aplacer, pointeur->fils_petit);
// utilisation de fonction réursive, appel a la fonction avec une nouvelle valeur
}
else
{
    if(pointeur->fils_grand == NULL)
    {
        pointeur->fils_grand=aplacer;
        aplacer->parent=pointeur;
    }
    else placer_noeud(aplacer, pointeur->fils_grand);
}
}

void affiche(NOEUD *saintpere)
{
    NOEUD *pointeur;
    pointeur=saintpere;

    if(pointeur->fils_petit != NULL) affiche(pointeur->fils_petit); // fonction réursive
    printf("%d\n",pointeur->valeur);

    if(pointeur->fils_grand != NULL) affiche(pointeur->fils_grand); // fonction réursive
}

```