TP C++

Section : M.I

En annexe vous trouver un programme C qui implémente un arbre père-fils faisant appel à une fonction de création et une fonction d'affichage pour des produits.

On souhaite développer une classe template pour gérer un arbre père-fils qui fonctionne pour des données de différents type.

Ecrire la classe template arbre. Prévoir :

- Un constructeur
- Un destructeur
- Une méthode d'insertion
- Une méthode d'affichage
- Une méthode de recherche
- Une méthode de suppression d'un élement de l'arbre

Tester à travers un programme C++ l'ensemble des méthodes implémenter.

Annexe

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


char  io[100];

struct produit
{
   int   id;
   char  name[100];
   float prix;
```

```
};
typedef struct produit PRODUIT;
int lire_produit(PRODUIT *d, int *key_pere);

struct arbre
{
   PRODUIT d;
   struct arbre *fils;
   struct arbre *frere;
};
typedef struct arbre ARBRE;
int insere(ARBRE **racine,PRODUIT d, int key_pere);

void creer_arbre(ARBRE **racine)
{
   PRODUIT d;
   int key_pere;

   *racine=NULL;

   while (lire_produit(&d,&key_pere))

     insere(racine,d,key_pere);

}

int insere(ARBRE **racine,PRODUIT d, int key_pere)
{
   ARBRE *p;

   if(*racine)
   {
      if( (*racine)->d.id==key_pere)
      {
         p=(ARBRE *) malloc(sizeof(ARBRE));
```

```
            p->d=d;
            p->fils=NULL;
            p->frere=(*racine)->fils;
            (*racine)->fils=p;
            return(1);
         }
         else
            if(insere(&(*racine)->frere,d,key_pere))
               return(1);
            else
              return(insere(&(*racine)->fils,d,key_pere));
      }
      else
      {
         if(key_pere==0)
         {
          p=(ARBRE *) malloc(sizeof(ARBRE));
          p->d=d; p->fils=NULL; p->frere=*racine; *racine=p;
          return(1);
         }
         else
            return(0);
      }
}

int lire_produit(PRODUIT *d, int *key_pere)
{
   printf("Id       :");gets(io); sscanf(io,"%d",&(d->id));
   if(!d->id) return(0);
   printf("Nom      :");gets(d->name);
   printf("Prix     :");gets(io); sscanf(io,"%f",& (d->prix));
   printf("Cle Pere  :");gets(io);sscanf(io,"%d",key_pere);

   return(1);
}
```

```
void view_arbre(ARBRE *racine)
{
   if(racine)
   {
      printf("\n%d %s %10.3f", racine->d.id, racine->d.name, racine-
>d.prix);
      view_arbre(racine->frere);
      view_arbre(racine->fils);
   }
}

int main()
{
   ARBRE *racine;


   creer_arbre(&racine);
   view_arbre(racine);
   return 0;
}
```