Chapitre 8 : Le polymorphisme

Le polymorphisme forme avec l'encapsulation et l'héritage les trois piliers des langages orientés objets.

1. Redéfinition des fonctions

Exemple

Utilisation d'un pointeur de base pour manipuler un objet d'une classe dérivée.

```
class materiel
{
     protected :
          char ref[20];
          char marque[20];
     public :
          materiel(char *r, char *m)
          {
             strcpy(ref,r);
             strcpy(marque,m);
          }
      void affiche()
      {cout<<"Reference : "<<ref<<"\tMarque : "<<marque<<endl;}
};

//-------------------------------------------------------------

class micro:public materiel
{
       char processeur[20];
       int  disque;
     public :
       micro(char *r,char *m,char *p,int d):materiel(r,m)
          {
            strcpy(processeur,p);
            disque=d ;
          }
     void affiche()
     {
          cout<<"Reference : "<<ref<<"\tMarque : "<<marque
          <<"\tProcesseur : "<<processeur
          <<"\tDisque : "<<disque<<endl;
     }
};
//-----------------------------------
```

```
main ()
{
     materiel  *p;
     p=new materiel("HP5010","HP");
     p->affiche();
     p=new micro("IBM7","IBM","P4",40);
     p->affiche();
     delete p;
}
```

Exécution

```
Reference : HP5010       Marque : HP
Reference : IBM7         Marque : IBM
```

Commentaire

- Le pointeur 'p' est utilisé pour stocker l'adresse d'un objet de la classe 'materiel', appeler la fonction 'affiche' et détruire l'objet créé.

- Le pointeur 'p' est ensuite utilisé pour réaliser les mêmes opérations sur un objet de la classe 'micro'.

-Dans le cas des deux objets créés, c'est la méthode 'affiche' de la classe 'matériel' qui a été utilisée. Le compilateur a donc tenu compte du type de pointeur (matériel*) et non du type d'objet pointé par 'p'.

- L'idéal serait donc que le compilateur appelle 'affiche' de 'micro' quand 'p' pointe sur 'micro' et 'affiche' de 'matériel' quand 'p' pointe sur 'matériel'.

- Pour obtenir ce résultat, il suffit d'indiquer au compilateur que la fonction 'affiche' est une fonction polymorphe, c.à.d une fonction virtuelle.

Définition

Une fonction polymorphe (ou virtuelle) est une méthode qui est appelée en fonction du type d'objet et non en fonction du type de pointeur utilisé.

Exemple

Pour que le programme précédent fonctionne correctement, il faut déclarer la fonction 'affiche' une fonction polymorphe ou virtuelle en précédant son prototype par le mot clé 'virtual'.

```
class materiel
{
     protected :
          char ref [20+1];
          char marque [20+1];
     public :
          materiel (char *r, char *m)
          {
            strcpy(ref,r);
            strcpy(marque,m);
          }
      virtual void affiche()
      {cout<<"Reference : "<<ref<<"\tMarque : "<<marque<<endl;}
};
class micro : public materiel
{
       char processeur [20+1];
       int  disque;
     public :
       micro(char *r,char *m,char *p,int d): materiel(r,m)
          {
            strcpy(processeur,p);
            disque=d ;
          }
     void affiche()
     {
          cout<<"Reference : "<<ref<<"\tMarque : "<<marque
          <<"\tProcesseur : "<<processeur
          <<"\tDisque : "<<disque<<endl;
     }
};
//---------------------------------------
main ()
{
     materiel  *p;
     p=new materiel("HP5010","HP");
     p->affiche();
     p=new micro("IBM7","IBM","P4",40);
     p->affiche();
     delete p;
}
```

Exécution

```
Reference : HP5010      Marque : HP
Reference : IBM7        Marque : IBM    Processeur : P4 Disque : 40
```

Remarque

Seule la fonction 'affiche' de la classe de base 'matériel' doit être déclarée virtuelle. Cependant, il est conseillé d'indiquer le mot clé 'virtual' pour toutes les fonctions concernées par le polymorphisme, et cela, quel que soit leur niveau dans la hiérarchie des classes.

Exercice

Ajouter une classe 'imprimante' et une classe 'scanner' dérivées de la classe 'matériel' et écrire un programme global sous forme de menu donnant à l'utilisateur la possibilité de choisir le type de matériel à afficher.

Remarque

- Lorsqu'on appelle une fonction virtuelle pour un objet, le C++ cherche cette méthode dans la classe correspondante. Si cette fonction n'existe pas dans la classe concernée, le C++ remonte la hiérarchie des classes jusqu'à ce qu'il trouve la fonction appelée.

- Une fonction virtuelle ne peut être appelée par une autre méthode de la classe.

- Une fonction virtuelle ne peut être utilisée dans le corps d'un constructeur ou d'un destructeur.

2. Fonction virtuelle pure et classe abstraite

2.1. Fonction virtuelle pure

- Une fonction virtuelle pure est une fonction telle que par exemple :

```
virtual void affiche()=0;
```

- Définir une fonction virtuelle pure entraîne :

- Une classe qui contient la définition d'une fonction virtuelle pure devient une classe abstraite.
- La redéfinition des fonctions virtuelles pures est obligatoire dans toutes les classes dérivées. Dans le cas contraire, les classes dérivées deviennent obligatoirement abstraites.

2.2. Classes abstraites

- Une classe abstraite est une classe qui ne peut être instanciée. Par contre, on peut créer des objets de ce type via l'héritage.

Le programme complet :

```
class materiel
{
      protected :
          char ref [20+1];
          char marque [20+1];
     public :
          materiel (char *r, char *m)
          {
            strcpy(ref,r);
            strcpy(marque,m);
          }
        virtual void affiche()=0;
};
void materiel::affiche()
{cout<<"Reference : "<<ref<<"\tMarque : "<<marque<<endl;}
//-----------------------------------------------------
class micro : public materiel
{
        char processeur [20+1];
        int  disque;
     public :
       micro(char *r,char *m,char *p,int d): materiel(r,m)
          {
            strcpy(processeur,p);
            disque=d ;
          }
     void affiche()
     {
          cout<<"Reference : "<<ref<<"\tMarque : "<<marque
          <<"\tProcesseur : "<<processeur
          <<"\tDisque : "<<disque<<endl;
        }
};
//-----------------------------------------
main()
{
    materiel   *p;
    //p=new materiel("HP5010","HP");
    //p->affiche();
    p=new micro("IBM7","IBM","P4",40);
    p->affiche();
    delete p;
}
```