partie 5 : Méthode virtuelle et classe abstraite

MOOC Langage C++ INSA

Source : Ce document est en partie extrait du livre : du langage C au C++ par Thierry Monteil, Vincent Nicomette, François Pompignac, Saturnino Hernando, Presses Universitaires du Midi ISBN 978-2-8107-0054-7

1 Membres et classe virtuelle

Un opérateur virtuel (`virtual type nom_fonction(..)`) a la caractéristique de pouvoir être redéfini dans les classes dérivées. La nouvelle définition est utilisée chaque fois que l'objet appelant est de type dérivé. Ceci met en jeu la liaison dynamique. Dans ce cas, l'appel de la fonction est lié à sa définition lors de l'exécution. Ceci s'oppose à la liaison statique où l'appel de la fonction est lié lors de la compilation.

Exemple:

```
class Base {
     public:
          int non_virtuelle() const {return 1;}
          virtual int virtuelle() const {return 1;}
          int base_nonvirtuelle(){return non_virtuelle();}
          int base_virtuelle(){return virtuelle();}
};
// fonctions externes
int externe_nonvirtuelle(Base &b){return b.non_virtuelle();}
int externe_virtuelle(Base &b){return b.virtuelle();}
//derivee
class Derive : public Base {
     public:
          int non_virtuelle() const {return 2;}
          virtual int virtuelle() const {return 2;}
};
//main
int main (){
     Derived d;
     //la liaison statique pose Pb ici
     cout << externe_nonvirtuelle(d) << endl; /* affiche 1 : appel
     la méthode de base */
     cout << d.base_nonvirtuelle() << endl;/* affiche 1 : appel
     la méthode de base */
     //ALORS que la liaison dynamique le résoud
     cout << externe_virtuelle(d) << endl; /* affiche 2 : appel
```

```
    bien la méthode de derive */
    cout << d.base_virtuelle() << endl;/* affiche 2 : appel
    bien la méthode de derive */
    return 0;       }
```

Dans l'exemple ci-dessus, on remarque que, pour la fonction définie comme virtuelle, c'est bien la dernière définition dans la classe dérivée qui est utilisée. Pour la fonction non virtuelle, on a une liaison statique et c'est la définition de la classe de base qui est prise.

Remarques:

— Ce mécanisme de liaison dynamique permet de résoudre certains problèmes, toutefois il ralentit les programmes. C'est donc à éviter quand on a des contraintes de temps d'exécution,

— `virtual` est aussi utilisé dans le cas de l'héritage multiple afin d'éviter d'hériter plusieurs fois d'une même classe et donc de dupliquer ses attributs.

2 Membre et classe abstraite

En conception orientée objet, il existe une famille de méthodes ou de classes un peu particulière. Leur rôle n'est pas d'être appelé ou instancié directement dans un programme. Elles sont là pour rappeler au programmeur qu'il faudra les définir correctement à un moment. On dit qu'une classe est abstraite à partir du moment où elle contient une méthode abstraite. Une méthode abstraite est une méthode où l'on a défini son nom et ses arguments sans donner son code. Ce sont ses filles qui donneront le code des méthodes lors du mécanisme d'héritage. C'est un moyen pour forcer des classes filles à posséder un certain nombre de méthodes, quand on ne sait pas encore dans la classe mère comment écrire cette méthode. La syntaxe est la suivante : **Exemple:**

```
class polygone{
    ...
    virtual int perimetre() =0;
    ...
};
class carre : public polygone{
    int cote;
    ...
    virtual int perimetre(){ return (4*cote);}
    ...
};
```

3 Identification de type à l'exécution

Il existe un opérateur nommé `typeid` qui prend en paramètre un pointeur ou une référence sur un objet et qui renvoie un objet de type `type_info`. Ce dernier possède une méthode `name()` qui donne le nom de la classe de l'objet précédemment passé en paramètre pour un compilateur fixé. Cette classe possède aussi les opérateurs de comparaison == et ! =.

```
class point {
...
};
int main(){
   point p;
```

```
   point *ap;
   ap=&p;
   cout << typeid(ap).name()<<endl; // affiche: point
   return 0;
}
```

Ces outils sont utilisés lors de l'exécution du programme quand le polymorphisme est utilisé afin de savoir réellement la classe dont est issu un objet particulier.