CHAPITRE 10

Identification et spécification des opérations - II

Méthodes virtuelles vs. non-virtuelles

Afin de conserver la compatibilité du langage C++ avec les langages antérieurs, particulièrement le C, les concepteurs du langage C++ ont défini deux types de méthodes:

- Les méthodes non-virtuelles (type par défaut),
- Les méthodes virtuelles.
  - Le C++ est l'un des rares langages orientés-objet qui permette de contrôler le type de convention d'appel qui sera utilisé pour chaque méthode d'une classe.
  - La majorité des langages orientés-objet n'ont que des fonctions virtuelles (Python, Java, Eiffel).

Méthodes virtuelles vs. non-virtuelles

Les conventions d'appel des méthodes virtuelles et non-virtuelles sont très différentes:

- L'appel à une méthode **non-virtuelle** est traité de façon statique: un appel à un symbole fixe est généré par le compilateur; ce symbole est résolu une fois pour toutes par l'éditeur des liens (*aiguillage statique* ou « *early binding* »),
- L'appel à une méthode **virtuelle** ne sera résolu qu'au moment de l'exécution, et dépendra de la classe de l'objet sur lequel la méthode sera effectivement invoquée (*aiguillage dynamique* ou « *late binding* »).

Méthodes virtuelles vs. non-virtuelles

```
#include <iostream>

class Hello
{
public:
   void imprimer( ostream&, const char* );
   virtual void v_imprimer( ostream&, const char* );
};

void Hello::imprimer( ostream& o, const char* s)
{
   o << s << endl;
}

void Hello::v_imprimer( ostream& o, const char* s)
{
   o << s << endl;
}

int main( int argc, char** argv )
{
   Hello* h = new Hello;
   h->imprimer( cout, "fonction non-virtuelle" );
   h->v_imprimer( cout, "fonction virtuelle" );
}
```

Méthodes virtuelles vs. non-virtuelles

Code assembleur généré pour chacune des lignes suivantes:

```
   h->imprimer( cout, "fonction non-virtuelle" );
   h->v_imprimer( cout, "fonction virtuelle" );
```

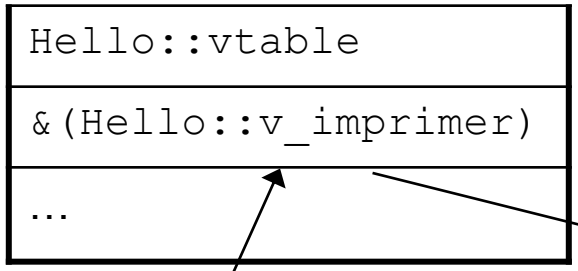
```
.LM12:
        addl $-4,%esp
        pushl $.LC0
        pushl $cout
        movl -4(%ebp),%eax
        pushl %eax
        call imprimer__5HelloR7ostreamPCc
        addl $16,%esp
.stabn 68,0,24,.LM13-main
.LM13:
        addl $-4,%esp
        movl -4(%ebp),%edx
        movl (%edx),%eax
        addl $8,%eax
        pushl $.LC1
        pushl $cout
        movl -4(%ebp),%edx
        pushl %edx
        movl (%eax),%esi
        call *%esi
        addl $16,%esp
```

symbole résolu par l'éditeur de liens

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Appel à une méthode virtuelle

Dans le cas d'une méthode virtuelle, l'appel se fait de façon indirecte, par le biais d'une table de pointeurs sur les fonctions virtuelles définie pour la classe (*vtable*):




```
.LM13:
    addl $-4,%esp
    movl -4(%ebp),%edx
    movl (%edx),%eax
    addl $8,%eax
    pushl $.LC1
    pushl $cout
    movl -4(%ebp),%edx
    pushl %edx
    movl (%eax),%esi
    call *%esi
    addl $16,%esp
```

```
.LM7:
    addl $-8,%esp
    pushl $endl__FR7ostream
    addl $-8,%esp
    movl 16(%ebp),%eax
    pushl %eax
    movl 12(%ebp),%eax
    pushl %eax
.LCFI9:
    call __ls__7ostreamPCc
    addl $16,%esp
    movl %eax,%eax
    pushl %eax
    call __ls__7ostreamPFR7ostream_R7ostream
    addl $16,%esp
```

La table des méthodes virtuelles d'une classe

La table des méthodes virtuelles d'une classe est construite:

- À partir de la table de méthodes virtuelles de sa classe de base,
- En remplaçant les adresses des méthodes virtuelles explicitement surchargées dans la classe par les adresses des nouvelles définitions,
- En ajoutant, à la fin de la table, les adresses des nouvelles méthodes virtuelles définies pour la classe.

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

La table de méthodes virtuelles

```
class Vehicule
{
public:
   virtual void rejoindre(Destin&) = 0;
};

class Avion : public Vehicule
{
public:
   virtual void rejoindre(Destin&);
   virtual void decoller(void);
   virtual void atterrir( void ) = 0;
};

class CL_415 : public Avion
{
public:
   virtual void rejoindre(Destin&);
   virtual void aterir( void );
};
```

Vehicule::vtable
NULL

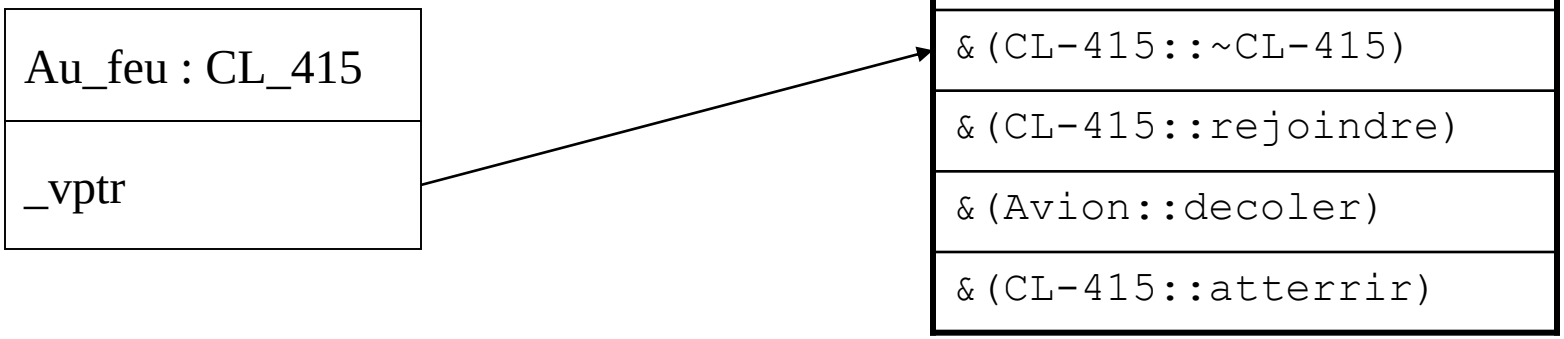
Avion::vtable
&(Avion::rejoindre)
&(Avion::decoller)
NULL

CL-415::vtable
&(CL-415::rejoindre)
&(Avion::decoller)
&(CL-415::atterrir)

Mécanisme d'appel à une méthode virtuelle

Chaque objet (instance) C++ appartenant à une classe qui contient au moins une méthode virtuelle, contient un pointeur caché (*vptr*) qui permet au compilateur de retrouver la table des méthodes virtuelles de la classe.

```
CL_415* au_feu = new CL_415;
```



Mécanisme d'appel à une méthode virtuelle

Des instructions sont ajoutées (automatiquement par le compilateur) à la fin du constructeur pour **initialiser** le *vptr*.

```
Vehicule::Vehicule() {
    vptr = &Vehicule::vtable; // Code ajouté automatiquement
    [...]                     // Corps du constructeur
}


Avion::Avion()
   : Vehicule() {

    vptr = &Avion::vtable; // Code ajouté automatiquement
    [...]                  // Corps du constructeur
}

CL_415::CL_415()
   : Avion() {

    vptr = &CL_415::vtable; // Code ajouté automatiquement
    [...]                   // Corps du constructeur
}
```

Mécanisme d'appel à une méthode virtuelle

Le compilateur connaît l'offset de la méthode dans la *vtable*. L'adresse de la méthode à appeler est donc calculée lors de l'exécution en prenant l'adresse de base contenue dans le *vptr* additionnée à l'offset de la méthode dans la *vtable*.

```
//L'appel au constructeur de la classe CL_415 fait en sorte
//que le vptr de v pointe sur la vtable de CL_415.
Vehicule* v = new CL_415();  //v->vptr = &CL_415::vtable

//À partir de l'adresse de base de la vtable de CL_415 et
//de l'offset de la fonction rejoindre() dans la vtable,
//la méthode CL_415::rejoindre() sera appelée.
v->rejoindre();  //CL_415::rejoindre()
```