

PROGRAMMATION C ++

MODELISATION OBJET

Interface : entête des fonctions, prototype

L'implémentation : organisation des données et définition des fonctions

STRUCTURE ET CLASSE

En C :

- **.h** : fichier → publication
Définition des types (structures)
Prototype des fonctions exportées
- **.c** → non publié
Définition de fonction (implémentation)

En C++ :

- **Classes** : type d'objet
Un objet est une instance de classe.

CLASSE

Les instances partagent :

- La structure de leurs données
- Les méthodes (fonctions associées aux classes)
- Les objets partagent leur comportement

POLYMORPHISME

Les différentes classes partagent une interface mais **pas l'implémentation** → Deux classes peuvent être utilisées avec des méthodes de même nom → **Surcharge de fonction**.

HERITAGE

Classe dérivé : hérite de la structure / comportement de la base.

→ Enrichissement de la base

- Nouvelles informations
- Redéfinir des méthodes
- Étendre des méthodes

→ A quoi ça sert ?

- Factoriser le code
- Contrat d'interface
- Module de fonctionnalités génériques
- Ajout
- Tester

LES CLASSES EN C++

CLASSE

Etat : **attribut** (champs de structure)

Comportement : **méthode** (fonction membre)

— —: Membres de la classe

Accessibilité et d'accès aux membres : public, private, protected (dérivée seulement)

Définition de la classe :

```
// fichier instru.cpp
#include "instru.h "
#include <iostream>
```

```
string Instru::getName() {
    return _name;
}
```

```
void Instru::setName(string name) {
    _name = name;
}
```

```
void Instru::print() {
    cout << getName();
}
```

On doit préciser que c'est la méthode **getName** de la classe **Instru**

:: est l'opérateur de **résolution de portée**

Les méthodes accèdent aux membres de la classe et s'appliquent implicitement à la même **instance**

Qualificateur const : **interdit la modification** du paramètre, peut être utilisé par une méthode et il n'y a pas de modification de l'instance.

Pointeur this : Lien entre méthode/objet

This adresse de l'objet qui appelle la méthode.

→ Lève l'**ambiguïté sur un nom de méthode**

→ Distinction entre attribut et variable locale.

CONSTRUCTEUR

Méthode : Le constructeur est appelé lors de l'instanciation d'un objet

Même nom de classe

Pas de type retour

Plusieurs paramètres (aucun par défaut)

Peut être surchargé

Constructeur par recopie :

```
Instru i = anotherInstruObject; // ceci n'est pas une  
affectation !
```

Appel au constructeur par recopie dont le prototype est :

```
Instru(const Instru &);
```

DESTRUCTEURS

Méthode particulière qui est appelée lorsque l'objet arrive en fin de durée de vie (variable locale à la fin d'une fonction par exemple).

- **Porte le nom de la classe précédé de ~**
- **Pas de type de retour**
- **Pas de paramètres**

Appelé implicitement en fin de vie de variable, ou par **delete** et **delete[]** .

Utile pour libérer des ressources allouées dynamiquement.

ACCESSIBILITE

Les constructeurs ou destructeurs peuvent être **public**, **protected** ou **private**

- ❖ Interdiction de créer une instance directement (on utilisera des méthodes **static**)
- ❖ Interdiction de détruire une instance avec **delete**

MEMBRES STATIQUES

Chaque attribut est par défaut associé à une instance de la classe : il s'agit alors d'un **attribut d'instance**

Certains attributs peuvent être associés à la classe directement :

Ce sont des **attributs de classe**, initialisés une seule fois.

Utilité : compter le nombre d'instances, contrôler les créations d'instance

HERITAGE ET POLYMORPHISME

- *Composition*: Crée une classe en fonction d'une autre classe.
- *Association* :
- *Héritage* : Les classes ont des aspects en commun, mais également des aspects spécifiques.
 - Aspects en commun (attributs)
 - Comportements (méthodes)

Une classe dérivée hérite automatiquement des attributs et des méthodes de la classe de base.

Il y a différents types d'accessibilités (private, public, protected).

Protected_: visible seulement par les classes qui vont hériter.

DECLARATION ET INSTANCIATION

Sans pointeur : L'objet est déclaré et j'instancie en même temps.

Avec pointeur : L'objet est déclaré mais je n'instancie pas. Pour instancier, on utilise une classe dérivée.

Sans polymorphisme : le contenu se comporte comme un objet de classe `*nom_de_la_classe*` (Déclaration)

Avec polymorphisme : le contenu se comporte comme un objet de la classe dérivée

`*nom_de_la_classe*`(Instanciation)

LES FONCTIONS VIRTUELLES

Une méthode qualifiée de *virtual* est une **méthode à liaison tardive** (elle bénéficie du polymorphisme).

Un constructeur ne peut pas être *virtual*.

Méthodes virtuelles pures : redéfinies et étendues dans les classes dérivées

(Une classe peut comporter des méthodes virtuelles « classiques » (avec une implémentation) et des méthodes virtuelles pures.)

LES CLASSES ABSTRAITES

Pas de code fourni : donc pas d'appel possible à cette méthode

LE POLYMORPHISME EN ACTION

exemple cours