الكلية متعددة التخصصات تارودانت
ⵜⴰⴽⵍⵉⵜ ⵜⴰⵙⵙⵉⵙⵉⵜ ⵏ ⵜⵓⵙⵎⵓⵔⵜ
FACULTE POLYDISCIPLINAIRE TAROUDANT

Programmation Orientée Objet avec C++

Filière Génie Informatique Semestre 4

Pr Y. ES-SAADY
y.essaady@uiz.ac.ma

2024/2025

Chapitre 3

Les fonctions amies

Y. Es-saady

Les fonctions amies

- En principe, l'encapsulation interdit à une fonction membre d'une classe ou toute fonction d'accéder à des données privées d'une autre classe.
- Mais grâce à la notion d'amitié entre fonction et classe, il est possible, lors de la définition de cette classe d'y déclarer une ou plusieurs fonctions (extérieurs de la classe) amies de cette classe.
- Une telle déclaration d'amitié les autorise alors à accéder aux données privées, au même titre que n'importe quelle fonction membre. Il existe plusieurs situations d'amitiés :
  - **1. Fonction indépendante, amie d'une classe.**
  - **2. Fonction membre d'une classe, amie d'une autre classe.**
  - **3. Fonction amie de plusieurs classes.**
  - **4. Toutes les fonctions membres d'une classe, amies d'une autre classe.**

Fonction indépendante amie d'une classe

- Pour déclarer une fonction amie d'une classe, il suffit de la déclarer dans cette classe en la précédent par le mot clé **'friend'.**
- **Exemple:**

```
class point{
private:
   int x,y;
public :
point(int a=0,int b=0)
   {x=a; y=b;}
friend bool
coincide(point,point);
};
```

```
bool coincide(point p1,point p2){
   if(p1.x==p2.x && p1.y==p2.y) return
true; else return false;
}
main(){
   point o1(15,2), o2(15,2),
if(coincide(o1,o2)) cout<<"les objets
coïncident\n"; else cout<<"les objets
sont différents\n";
}
```

Fonction indépendante amie d'une classe

- La fonction «**coincide**» est une fonction amie de la classe « point».
- Déclarer la fonction «**coincide**» « private » ou « public » importe peu.
- La fonction **coincide** est une fonction amie et **non pas une fonction membre de la classe point.**

```
// Fausse (déclaration utilisée pour une fonction membre).
bool point::coincide(point p1,point p2){
……..;
}
// Correcte (déclaration utilisée pour une fonction amie).
bool coincide(point p1,point p2){
            …….
}
```

Fonction membre d'une classe, amie d'une autre classe

- Faire attention à l'ordre des déclarations et des définitions.
  - la fonction membre **'f'** de la classe '**B**' peut accéder aux membres privées de n'importe quel objet de la classe 'A'.
  - Pour compiler correctement la déclaration de la classe 'A', contenant une déclaration d'une fonction amie, il faut :
    - **Définir 'B' avant 'A'.**
    - **Déclarer 'A' avant 'B'.**

```
class B;
class A {
    ……….
  public :
   friend int B::f(int,A);
};
//-----
class B {
……….
   public:
    int f(int,A);
};
//----------
int B::f(int,A) {
……….
}
```

Exemple: Fonction membre d'une classe, amie d'une autre classe

```cpp
class B; // Declaration anticipee

class A {
public:
    void afficherB(const B& obj); // Methode membre qui sera amie de B
};

class B {
private:
    int valeur;

public:
    B(int v) : valeur(v) {}

    // Declaration de la fonction membre de A comme amie
    friend void A::afficherB(const B& obj);
};

// Definition de la methode membre de A qui accede aux membres prives de B
void A::afficherB(const B& obj) {
    cout << "Valeur privee de B : " << obj.valeur << endl;
}

int main() {
    A a;
    B b(42);

    a.afficherB(b); // Acces a la valeur privee de B via A

    return 0;
}
```

Fonction amie de plusieurs classes

- Toute fonction membre ou indépendante, peut être amie de plusieurs classes.

```
class A
{
.....
public :
friend void f(A,B);
};//----------------------------
class B
{
.....
public :
friend void f(A,B);
};//-------------------------
void f(A,B)
{
.....
}
```

Toutes les fonctions membres d'une classe, amies d'une autre classe

- Au lieu de faire autant de déclarations de fonctions amies qu'il y a de fonctions membres, on peut résumer toutes ces déclarations en une seule.
- **Exemple**
  - **friend class B;** déclarée dans la classe 'A' signifie que toutes les fonctions membres de la classe 'B' sont amies de la classe 'A'.
- **Remarque**
  - Pour compiler la déclaration de la classe 'A', il suffit de la faire précéder de : 'class B ;' Ce type de fonction évite d'avoir à déclarer, les entêtes des fonctions concernées par l'amitié.

```
class A;
class B {
    public:
    void change (A&);
    void affiche(A&);
};
class A {
    int i;
    friend class B;
    public:
        A(int e=100) {i=e;}
};
void B::change(A& a) {
    a.i = 200; }
void B::affiche(A& a) {
cout << a.i << endl;
}
```

Toutes les fonctions membres d'une classe, amies d'une autre classe

```cpp
#include <iostream>
using namespace std;

class A; // Declaration anticipee

class B {
public:
    void change(A&);
    void affiche(A&);
};

class A {
private:
    int i;

    // Declaration de B comme classe amie
    friend class B;

public:
    A(int e = 100) { i = e; }
};

// Definition des methodes de B
void B::change(A& a) {
    a.i = 200;
}

void B::affiche(A& a) {
    cout << "Valeur de i : " << a.i << endl;
}
```

```cpp
int main() {
    A objA; // Valeur par defaut 100
    B objB;

    cout << "Avant modification :" << endl;
    objB.affiche(objA); // Affichage avant modification

    objB.change(objA); // Modification via la classe amie

    cout << "Apres modification :" << endl;
    objB.affiche(objA); // Affichage apres modification

    return 0;
}
```

```
Avant modification :
Valeur de i : 100
Apres modification :
Valeur de i : 200


--------------------------------
```

Remarques sur les fonctions amies

- **L'amitié n'est ni symétrique ni transitive**
  - Si la classe A est amie de la classe B n'implique pas la classe B est amie de la classe A (pas de symétrie)
  - Si la classe A est amie de la classe B **ET si la classe B est amie de la classe C** n'implique pas la classe A est amie de la classe C (pas de transitivité).

- **Quand utiliser l'amitié ?**
  - Dans le cas de l'amitié décrite dans "fonction indépendante, amie d'une classe", assez souvent l'amitié est utilisée quand la fonction manipule plus de deux objets à la fois.