La programmation orientée objet

Deux aspect : structurer les données et leur appliquer des traitements. Dans les chapitres précédents, vu la partie traitement : l'algorithmique. La POO vise à fournir un méthode pour structurer les données.

Classe et objet

Classe = type définit dans le code, objet = ce qui apparaît en mémoire dans le code. objet est l'instanciation d'une classe. Il peut y avoir plusieurs objets qui sont instancier à partir d'une classe (on peut également créer une classe mais ne pas instancier d'objet, mais l'intérêt est limité).

Déjà vu des exemples de classe : string, vector, array

Une classe rassemble des variables et des fonctions. S'appellent variables membres ou attribues et fonctions membres ou méthodes. La syntaxe :

```
class nom_de_la_classe {
    // définitions des variables et fonctions
};
```

(attention au point-virgule après les crocher

Pour utiliser une classe, comme un type (comme vous faites avec string depuis le début) :

```
nom_de_la_classe nom_de_variable {};
```

Bien sûr, peut s'utiliser comme paramètre de fonctions ou template :

```
class A {};
```

```
void f(A a) {} // fonction qui prend un objet de type A
comme paramètre
A a;
f(a);

A f() {} // fonction qui retourne un objet de type A
A a = f();
auto a = f();

template<typename T>
void f() {}

f<A>(); // type A comme argument template
```

Visibilité des membres d'une classe

3 types de visibilité :

- public (publique) : le membre est visible depuis l'extérieur de la classe ;
- private (privé) : le membre est n'est pas visible depuis l'extérieur de la classe ;
- protected (protégé) : le membre est visible que depuis les classes dérivées.

Le troisième cas est lié à la notion d'héritage, sera vu dans les classes à sémantique d'entité.

Par exemple, une classe A avec un membre f() ou i. Pour appeler le membre, on utilise l'opérateur . (comme déjà fait avec begin et end par exemple) :

```
A a {};  // défini a
a.i = 0; // accès à membre i
a.f();   // accès à membre f()
```

Bien faire attention aux notions de variables et types. A est un type, il permet de déclarer une variable. a est une variable, on peut appeler . dessus.

Dans ce code, on utilise la classe A, on est l'extérieur de la classe. Comme on a accès aux membres, on a donc un accès publique. Avec un membre privé, essayer d'accéder depuis l'extérieur produit une erreur du compilateur :

Chapitre précédent	Sommaire principal	Chapitre suivant

Cours, C++