Langage C++

- CIR2 -

ISEN ALL IS DIGITAL! NANTES | yncréa

TP7 : Template

Nils Beaussé (nils.beausse@yncrea.fr)

Partie 1 : Point générique

1. Créer un nouveau projet dans lequel vous implémentez la classe Point du TP6 en utilisant, cette fois-ci, des types numériques génériques.
2. Proposer une fonction test dans le main.cpp pour avoir l'affichage ci-dessous :

```
Point(T,T) : 0x7ffffe214870
(6.9,2.4)
~Point() : 0x7ffffe214870
```

3. Remplacez UNIQUEMENT float par int dans le test déjà proposé. Que constatez-vous ?
4. Créer une fonction template pointTest permettant de ne pas mettre le type en dur dans l'instanciation du point
5. Tester cette fois-ci votre code de la manière suivante :

```
void pointTests(){

    pointTest<int>();
    pointTest<float>();
}
```

Partie 2 : Tableau générique

1. On va créer une classe capable de manipuler des tableaux de types génériques.
2. Les attributs de la classe devront être, au minimum :
    a. Un pointeur de type générique qui constituera le tableau.
    b. Un indice de type int capable de contenir la taille du tableau.
    c. Un pointeur vers la fin du tableau.
3. Les méthodes de la classe devront contenir :
    a. Un constructeur qui prend en argument la taille du tableau.
    b. Un constructeur de copie.
    c. Un destructeur.
    d. Un accesseur pour retourner le pointeur vers la fin du tableau sans pouvoir le modifier
    e. Un accesseur pour retourner le pointeur vers le début du tableau sans pouvoir le modifier
4. Comme on souhaite pouvoir parcourir les éléments sans que l'utilisateur n'ait accès à tout le tableau on va créer une classe itérateur qui contiendra un pointeur vers un élément du tableau, cette classe devra contenir :
    a. Un accesseur qui renvoi l'élément en cours en copie
    b. Un accesseur qui renvoi le numéro de l'élément en cours
    c. Un constructeur qui prend en paramètre un pointeur sur le tableau et la taille du tableau
    d. Un manipulateur qui prend en paramètre un décalage à réaliser sur l'itérateur.
5. Une fois l'itérateur mis en place, incluez le dans votre classe tableau de façon à ce que l'utilisateur puisse avoir accès à l'itérateur et donc aux éléments du tableau. Astuce : pour qu'une classe puisse avoir accès aux membres d'une autre classe on peut utiliser le mot clé **friend.**

Par exemple :

```
class B;
class A
{
   private :
     int a ;
     f() ;
   friend B ;
}
class B
{
    void h (A *p) {
        p->a = 0 ;
        p->f() ;
    };
};
```

(ici la classe B peut accéder aux éléments privé de la classe A car elle est « amie »)

On peut aussi déclarer une fonction spécifique de B amie :

friend void B::f() ;

Seule celle-là sera amie avec la classe A et pourra accéder aux éléments privés.

Si la classe est template, comme d'habitude on doit rajouter template devant :

template <…> friend class B;
(friend avec toutes les classes B possible)

friend class B <A,B,C>**;**
A l'inverse : friend avec une seule classe B spécifique, celle qui aura les types A,B,C.

6. Testez votre tableau générique en float, en int, puis en char.

Partie 3 (bonus) : Guerre de personnages

1. On souhaite définir une classe personnage capable d'infliger des dégât grâce à une arme à un autre personnage.
2. Définir trois classe armes simple (épée, katana, pistolet) qui contiendront un facteur de dégât et une fonction qui retourne ces dégâts (tape()).
3. Définir une classe générique «fourreau» qui pourra contenir l'une de ces armes (grâce aux templates) : le fourreau contiendra une méthode : dégaine_et_tape qui appellera la fonction tape().
4. Définir une classe personnage qui contiendra un fourreau générique et sera capable, via un constructeur template, d'initialiser d'autres paramètre du personnage (sa vie et son nom par exemple) non template.
5. Créer trois personnages contenant ces armes et disposant d'un nom et de point de vie.
6. Définir une méthode capable d'attaquer un personnage en lui retirant ses points de vie.

Tester votre programme avec trois personnages se tapant dessus aléatoirement (ordre aléatoire d'appel des trois personnages) et afficher le décompte des points de vie (via une méthode interne à la classe personnage) jusqu'à qu'il ne reste qu'un gagnant.