Chapitre 10 : Les templates

Les templates (appelées aussi modèles ou patrons) représentent une technique favorisant la réutilisation et la spécialisation des fonctions et des classes.

Les templates utilisent un type paramètré (ou fictif) qui représente le type de donnée qui sera choisi (ou instancié) à l'utilisation du modèle.

1. Les fonctions templates

Une fonction template permet de définir une fonction qui réalisera le même traitement pour des types de données différents (type de donnée paramétré qui sera connu au moment de l'appel de la fonction).

Syntaxe

```
template<arguments du template>
type fonction(arguments)
{
      ……….
}
```

Ou

```
template<arguments du template> type  fonction(arguments)
{
      ……….
}
```

Description

- 'arguments du template' : s'écrit 'class type1, type2, …' où type1, type2... représentent les types paramétrés (par convention on prend la lettre T, U …).
- La fonction déclarée peut utiliser ensuite le ou les types fictifs.

Exemple

Les fonctions surchargées suivantes :

```
void affiche(short v)
{ cout<<"Valeur : "<<v<<endl; }
//---------------------------
void affiche (float v)
{ cout <<"Valeur : "<<v<<endl; }
//-------------------------------
void affiche (double v)
{ cout <<"Valeur : "<<v<<endl; }
```

peuvent être remplacées par la fonction modèle suivante :

```
template <class  T>
void affiche (T v)
{ cout <<"Valeur : "<<v<<endl; }
```

Et on peut faire des appels tels que :

```
main()
{
      short      n=15;
      float      x=3.7;
      double     y=5.14;
      affiche(n);     affiche(34);
      affiche(x);     affiche(5.8);
      affiche(y);
      affiche('A');
      affiche("BONJOUR");
      getch() ;
}
```

Exécution

Remarque

Une fonction template peut employer plusieurs arguments :

```
template <class  T,class  U>
void affiche(T v1,U v2)
{
      cout <<"Valeur 1 : "<<v1<<endl;
      cout <<"Valeur 2 : "<<v2<<endl;
}
```

Exercice

Définir une fonction template ou modèle appelée 'max' dont l'objectif est de renvoyer le plus grand des deux éléments passés en argument.

2. Les classes templates

Syntaxe

```
template <arguments du template>
class nom_de_la_classe
{
      ……;
      ……;
};
```

Description

C++ impose pour définir le corps des fonctions membres d'employer la syntaxe suivante :

```
template <arguments du template>
type nom_de_la_classe<arguments>::fonction(arguments)
{
      ……….
      ……
}
```

- Une classe modèle ou template peut également contenir des fonctions membres qui n'utilisent pas les types paramétrés. Cependant, même dans ce cas, le corps de la fonction doit respecter la contrainte liée au nom complet de la fonction membre.

Exercice 1

Reprendre l'exemple de la classe 'tableau' et écrire un programme global à partir des déclarations suivantes :

```
template <class T>
class tableau
{
        int    ne;
        T      *p;
      public :
        tableau(int n){ p=new T[ne=n];}
        T & operator[](int n){ return p[n]; }
        ~tableau(){delete []p;}
};
//---------------------------------------
main()
{
      tableau<int> t1(5);
      for(int i=0;i<5;i++) t1[i]=i+1;
      for(int i=0;i<5;i++) cout<<t1[i]<<"\t"; cout<<endl;
      //-----------------------------------
      tableau<float> t2(7);
      for(int i=0;i<7;i++)t2[i]=i*2.3;
      for(int i=0;i<7;i++)cout<<t2[i]<<"\t"; cout<<endl;
      //-------------------------------------------------------
      tableau<char> t3(8);
      for(int i=0;i<8;i++)t3[i]='A'+i;
      for(int i=0;i<8;i++)cout<<t3[i]<<"\t"; cout<<endl;

      getch();
}
```

Exécution

Exercice 2

Redéfinir la classe 'point' pour des coordonnées de type paramétré (short, int ou long) et donner des exemples d'instanciation de points.

Programme

```
template <class U>
class  point
{
             U x;
             U y;
       public :
              point(U a,U b){x=a; y=b;}
              void affiche(){cout<<"X="<<x<<" - Y="<<y<<endl;}
};
//-------------------------------------------------------------
main ()
{
    point<int>    p1(10,20);                p1.affiche();
    point<double> p2(23.45,20.76);          p2.affiche();
    point<float>  p3(1.98,2.907);           p3.affiche();
    point<char>   p4('F','K');              p4.affiche();
}
```

Exécution