Université Ibn Tofail
جامعة ابن طفيل
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵉⴱⵏ ⵟⴼⴰⵢⵍ
Ibn Tofail University

POO sous C++
Patrons (Modèles)

Med. AMNAI

Filière SMI - S5

Département d'Informatique

19 janvier 2021

Plan

Plan

Plan

Plan

Introduction

- 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.

Patrons de fonctions

- Si tout **ce qui change est le type**, mais tout le reste est le même, les patrons de fonction sont la solution.
- Il s'agit d'un **modèle** que le compilateur utilise **pour créer des fonctions au besoin**.

Exemple

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

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

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

```
//-----------------------------------
template <class T>

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

Exemple (1)

```
//-----------exptemp.cpp--------------
template <class T>

void affiche (T v){
     cout << " Valeur : " << v << endl;
}


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");

return 0 ;
}
```

```
Valeur : 15
Valeur : 34
Valeur : 3.7
Valeur : 5.8
Valeur : 5.14
Valeur : A
Valeur : BONJOUR
```

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;
}
```

Exemple (3)

Dans chacune de ces deux cas, quelle est la fonction qui sera appelée ?

```
template <typename T>
void echanger (T& a, T& b) {
    T temp = a;
    a = b;
    b = temp;
    cout << "Patron de fonction :  a = "
         << a << ", b = " << b << endl;
}

void echanger (int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
    cout << "Fonction int : a = "
         << a << ", b = " << b << endl;
}
```

```
template <typename T>
void echanger (T& a, T& b) {
    T temp = a;
    a = b;
    b = temp;
    cout << "Patron de fonction :  a = "
         << a << ", b = " << b << endl;
}

void echanger (double& a, double& b) {
    double temp = a;
    a = b;
    b = temp;
    cout << "Fonction double : a = "
         << a << ", b = " << b << endl;
}
```

```
int i = 4, j = 5;
echanger(i, j);
```

Appel de fonctions

Lors de l'appel à une fonction, le compilateur cherche dans :

- Les **fonctions ordinaires** en cherchant une signature identique des types d'arguments.
- Les **patrons de fonctions** avec une signature identique.
- Les **fonctions ordinaires** en essayant les **conversions**.
- Sinon **erreur** de compilation.

Patrons de Classes

```
//----------exptemp2.cpp----------------------------
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;

return 0;
}
```

```
1       2       3       4       5
0       2.3     4.6     6.9     9.2     11.5            13.8
A       B       C       D       E       F   G           H
```

Exercice

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

Exercice (Sol)

```
//-------------------------------------------------------------------------
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();

return 0;
}
```

```
X = 10 - Y = 20
X = 23.45 - Y = 20.76
X = 1.98 - Y = 2.907
X = F - Y = K
```

Conteneurs STL

La **STL** (Standard Template Library) définit un ensemble de conteneurs

- Séquences :
  - **Vector** : Tableau dynamique + (push_back, pop_back)
  - **List** : Liste chainée
  - **Queue** (push, pop)
  - **Deque** (push_back, pop_back, push_front_pop_front)
  - **Stack** (push, pop)
  - **Priority_queue** (Queue avec ordre)
- Conteneurs associatifs :
  - **Set** : Ensembles
  - **Map** : Tableaux assiciatifs
  - **Multiset**
  - **Multimap**