Université Savoie Mont Blanc,
UFR SceM
INFO702, M1 INFO

October 24, 2023

# TD2: patrons de fonctions et classes

(Fonctions génériques, classes génériques, spécialisation)

**Question 1.** Fonction générique : référencer le maximum de 3 valeurs

On veut pouvoir écrire qqchose du genre:

```
double x = 5.7;
double y = 10.2;
double z = 9.4;
double& t = ref_max( x, y, z );
```

A l'exécution, la référence `t` référence la variable `y`. Comment écrire de manière générique une telle fonction ?

**Question 2.** Fonction générique : argument minimum d'un tableau ou d'une collection

```
// Déclaration d'une fonction non générique de calcul de l'adresse
// d'un élément de valeur minimale dans un tableau entre l'adresse deb et
// l'adresse fin.
int* argmin( int* deb, int* fin );
```

Transformez la fonction précedente en une fonction générique valable pour tout itérateur dans une collection. Quels sont les concepts requis ? Ecrivez ensuite ce qui manque pour que le code suivant compile et s'exécute correctement:

```
struct Hotel {
  std::string name;
  double prix_par_nuit;
};

...
Hotel hotels[ 1000 ];
... // on remplit notre tableau d'hotels (on suppose que c'est fait)
// On veut savoir le moins cher
const auto it = argmin( hotels, hotels + 1000 );
std::cout << "Le moins cher s'appelle : "
          << it->name << " et coûte " << it->prix_par_nuit
          << std::endl;
```

Est-ce que cette fonction est adaptable à autre chose qu'un tableau, par exemple une liste d'éléments ?

**Question 3.** STL set

Qu'affiche le code suivant ?

```
std::set< int > S( { 3, 12, 7, 1, 15 } );
for ( auto v : S ) std::cout << " " << v;
std::cout << std::endl;
```

On sait maintenant que S peut prendre un 2ème paramètre générique qui est un type permettant la comparaison de deux valeurs. Ce type `T` doit fournir une méthode de comparaison.

```
bool operator()( int i, int j ) const {  ... }
```

Ecrire le type `Greater` qui permet d'induire un ordre décroissant dans les éléments de l'ensemble. On s'en servirait ainsi:

```
std::set< int, Greater > S( { 3, 12, 7, 1, 15 } );
for ( auto v : S ) std::cout << " " << v;
std::cout << std::endl;
```

**Question 4.** Fonction générique : partition d'une collection

Ecrivez une fonction générique qui réalise la partition d'une collection entre éléments qui satisfont un prédicat $P$ (mis au début) et éléments qui ne satisfont pas ce prédicat. Son prototype serait:

```
template <typename Iterator, typename UnaryPredicate>
Iterator partition( Iterator deb, Iterator end, const UnaryPredicate& P );
```

Par exemple, on peut s'en servir ainsi

```
int t[10] = { 17, 4, 8, 5, 13, 15, 1, 20, 13, 12, 9 };
auto    p = partition( t, t+10, [] (int v) { return v <= 10; } );
//    t[] = { 4, 8, 5, 1, 9, 15, 17, 20, 13, 12, 13 };
//      p ------------------------^
```

Quels concepts sont requis pour les types `Iterator` et `UnaryPredicate` ?

Cette fonction est utilisé dans l'algorithme bien connu Quicksort. Pourquoi demande-t-on usuellement que l'Iterator puisse être "random-access" (i.e. on peut le décaler de $n$ d'un coup, ou calculer la distance entre 2 itérateurs).

**Question 5.** Fonction générique : échange puis inversion d'un intervalle

Ecrire la fonction générique `swap` qui échange le contenu de deux variables. Ecrire ensuite la fonction générique d'inversion de l'ordre des valeurs dans un intervalle donné par deux itérateurs.

**Question 6.** Itérateurs dans intervalle donné

On veut faire une classe qui représente une collection d'entiers successifs, par exemple 3, 4, 5, 6, 7, sorte d'équivalent de `range(3,8)` en python. On pourrait l'utiliser ainsi:

```
Range R(3,8,1); // de 3 inclus à 8 exclus par pas de 1
for (auto v : R) std::cout << ' ' << v;
```

Proposez une telle classe. Comment la rendre encore plus versatile en ajoutant une contrainte supplémentaire que les entiers doivent satisfaire un prédicat ?

**Question 7.** Classe générique : Tuple

On veut écrire une classe qui représente un nombre $N \leq 32$ fixé à l'instanciation d'objets du même type donné aussi à l'instanciation (une sorte de tableau de taille fixe). On fournira un opérateur [] pour accéder aux éléments, ainsi qu'une fonction `size`. On spécialisera le cas `Tuple<N,bool>` pour tout stocker sur un seul entier.

**Question 8.** Classe générique : liste doublement chaînée

On veut faire un équivalent de std::list, c'est-à-dire une collection d'éléments rangés dans un certain ordre, où insertion et suppression autour de n'importe quel élément prend un temps constant. On stockera chaque élément dans une structure node qui stocke une valeur, et deux pointeurs vers les noeuds précédents et suivants. De plus toute liste contient un noeud fictif, qui matérialise le noeud end() invalide. Proposez une implémentation d'une telle liste.

**Question 9.** Classe générique : Arbre binaire

On veut écrire une classe générique `Arbre` qui modélise un arbre binaire pouvant stocker des valeurs de type `T`. On souhaite pouvoir créer un arbre l'arbre vide, ainsi que l'arbre à 1 élément. L'arbre doit pouvoir donner sa taille. On fera une classe interne Iterator qui permet de désigner un noeud de l'arbre. On ajoutera des méthodes qui donne l'itérateur sur la racine de l'arbre, puis des méthodes qui, étant donné un itérateur pointant sur un noeud, permettent de récupérer/ajouter/modifier son fils gauche ou son fils droit.

```
template <typename T>
struct Arbre {
  // type interne Arbre<T>::Node
  struct Node {
    T     value;
    Node* left;
    Node* right;
  };
```

```
  ...
private:
  Node* root;
};
```

On doit pouvoir parcourir les éléments de l'arbre avec un itérateur. On choisira l'ordre préfixe pour avoir ensuite naturellement les éléments par ordre croissant dans un ABR. Adaptez l'itérateur pour l'équiper d'une méthode `next` et des opérateurs ++

Ecrire ensuite une class générique `ABR` qui désigne un arbre binaire de recherche. Comment utiliser la classe `Arbre` ci-dessus ? Quels sont les concepts nécessaires pour le type `T` des valeurs ? Rajoutez une méthode d'insertion d'une valeur spécifique à un ABR.

Proposez enfin une méthode de tri de collection quelconque basé sur l'utilisation d'un ABR.

**Question 10.** Foncteurs et lambda

On veut pouvoir éliminer dans un ensemble de valeurs les valeurs qui ne sont pas dans un intervalle $[a, b]$ donné. Ecrire une classe `IsInInterval` qui permette de faire cela. Par exemple, le code suivant devrait fonctionner:

```
double values[7] = { -3.5, 15.4, 6.4, -17.2, 2.1, 25.2, 30.2 };
IsInInterval filter( 5.0, 19.0 );
auto last = std::remove_if( values, values+7, filter );
for ( auto it = values; it != last; ++it )
  std::cout << " " << *it;
std::cout << std::endl;
// Affiche: 15.4, 6.4, 2.1
```

Proposez ensuite une solution où vous écrivez directement la lambda-expression.

Soit maintenant le bout de code suivant :

```
double values[7] = { -3.5, 15.4, 6.4, -17.2, 2.1, 25.2, 30.2 };
...
std::for_each( values, values+7, ... );
```

Modifiez-le pour qu'il calcule : (1) le nombre d'éléments positifs, (2) le maximum des éléments, (3) la moyenne des éléments, en utilisant des lambda-expressions.

**Question 11.** Shared pointer

On veut écrire un type générique représentant un pointeur vers un élément partagé par plusieurs objets. Le principe est de compter le nombre de smarts pointeurs qui pointent vers la même chose. Lorsque ce nombre descend à 0, l'objet est détruit automatiquement.

```
struct Object {
  Object( std::string n ) : name( n ) {}
  std::string whoAmI() const { return name; }
  std::string name;
};
...
{
  Shared<Object> p1 = new Object( "I am shared." ); // acquis par p1, ref_count = 1
  Shared<Object> p2 = p1; // p2 et p1 pointent vers le même objet, ref_count = 2
  Shared<Object>* ptr_p3 = new Shared<Object>( p2 ); // *ptr_p3, p2, et p1 pointent
       vers le même objet, ref_count = 3
  // p1 et p2 se comportent comme des pointeurs.
  std::cout << (*p1).whoAmI()
            << " == " << p2->whoAmI()
            << " == " << (*ptr_p3)->whoAmI() << std::endl;
  p1->name = "I am modified."; // modifie les 3 affichages suivants.
  std::cout << (*p1).whoAmI()
            << " == " << p2->whoAmI()
            << " == " << (*ptr_p3)->whoAmI() << std::endl;
  delete ptr_p3; // ref_count = 2
} // p1 et p2 sont détruits, ref_count = 0. L'objet est alors désalloué.
```

Ecrire une telle classe générique à l'aide d'un patron de classe.