Travail dirigé : Éléments pour les structures de données en C++

Objectif:

Le but de ce laboratoire est de mettre en pratique certains éléments utiles pour l'implémentation de structure de donnes en C++.

Travail à faire

1. Template

Créez une fonction modèle (Template) qui permet d'envoyer dans un flux le contenu déréférencé de pointeurs. Son prototype est le suivant:

```
std::ostream& print_pointer(std::ostream& os, const std::string& type_name, const
pointer_type& ptr)
```

Développer ensuite un programme qui l'utilise. On pourra avoir successivement l'affichage d'un pointeur d'entier, d'un pointeur d'objet string ou encore d'un pointeur pointant sur un objet d'une classe que vous aurez définie. Une classe personne par exemple, ayant deux attributs (nom et prénom). Cette classe devrait surcharger l'opérateur <<

Exemple :

Si on a l'appel suivant :

```
  affiche_pointeur (cout, "string", ptr_string) << "!" << endl;
```

on devrait avoir l'affichage :

```
<string pointeur 0x563d9ed2beb0 valeur=bonjour>!
```

2. Coplien

Créez une classe Adresse et une classe Personne qui respecte la règle des 5 de Coplien.

L'interface de la classe Adresse vous est fournie ainsi que celle de la classe Personne (à compléter). Vous pouvez tester vos classes avec le programme testPersonne.cpp dont l'exécution devrait donner :

```
autre : Nom : bill, Prénom : Boquet
Adresse : 0x56545dd30eb0 ailleurs
moi : Nom : Joe, Prénom : blo
Adresse : 0x56545dd312f0 ici
copie
lui : Nom : Joe, Prénom : blo
Adresse : 0x56545dd31320 ici
constructeur déplacement
Nom : Joe, Prénom : blo
Adresse : 0x56545dd31320 ici
Nom : Joe, Prénom : blo
Adresse : 0x56545dd312f0 ici
```

3. lambda

Créez une classe `Etudiant` qui stocke le nom et les points d'un élève. Créez un tableau d'élèves et utilisez-le `std::max_element` pour trouver l'élève avec le plus de points, puis imprimez le nom de cet élève. `std::max_element` prend le `begin` et `end` d'une liste, et une fonction qui prend 2 paramètres et renvoie `true` si le premier argument est inférieur au second.

Étant donné le tableau suivant:

```
std::array<Etudiant, 8> resultats {
    {
      { "Albert", 3},
      { "Ben", 5},
      { "Christine", 2},
      { "Dan", 8}, //Dan a le plus de points (8)
      { "Enchilada", 4},
      { "Francis", 1},
      { "Greg", 3},
      { "Hagrid", 5}
    }};
```

votre programme devrait afficher :

```
Dan est le meilleur élève
```

4. lambda

Utilisez `std::sort` et un lambda dans le code suivant pour trier les saisons par température moyenne croissante.

```
#include <algorithm>
#include <array>
#include <iostream>
#include <string>

class Saison
{
public:
  Saison (const std::string& p_nom, double p_temperature)
  : m_nom (p_nom), m_temperatureMoyenne (p_temperature) { };
  //private:
  std::string m_nom;
  double m_temperatureMoyenne;
};

int
main ()
{
  std::array<Saison, 4> saisons {
    {
      { "Printemps", 285.0},
      { "Été", 296.0},
      { "Automne", 288.0},
      { "Hiver", 263.0}
    }};

/*
 * utiliser sort ici
*/

  for (const auto& saison : saisons)
    {
      std::cout << saison.m_nom << '\n';
    }
  return 0;
}
```

Le programme devrait afficher :

```
Hiver
Printemps
Automne
Été
```