Université de Nice-Sophia Antipolis — POLYTECH
ELEC4 — 2023–2024

Examen de C++

Durée : 1h00
Aucun document autorisé
Mobiles interdits

▶ 1. Expliquez de façon claire la différence entre le modèle de *programmation procédurale* et celui de *programmation objet*.

---

voir cours

.....

▶ 2. Expliquez de façon claire les notions de *constructeur* et de *destructeur* du langage C++.

---

voir cours

.....

▶ 3. Programmez la classe `date` formée :
- de trois variables menbres privées `jour`, `mois`, et `année` de type **int** ;
- d'un constructeur public (on ne vérifiera pas la validité de la date) ;
- de la méthode publique `toString` qui renvoie une chaîne de caractères au format *jj/mm/aaaa* ;
- de la surcharge de l'opérateur `<<` pour écrire une date dans un `std::ostream`.

```
pragma once

#include <string>

class date {
private:
  int jour, mois, année;

public:
  date(const int j, const int m, const int a) : jour(j), mois(m), année(a) {}

  std::string toString() const {
    return std::to_string(this->jour) + "/" + std::to_string(this->mois) +
      "/" + std::to_string(this->année);
  }

  friend std::ostream& operator<<(std::ostream &f, const date  &d) {
    return f << d.toString();
  }
};
```

La version de `toString` précédente ne respecte pas tout à fait l'énoncé puisqu'elle ne produit

pas nécessairement la chaîne *jj/mm/aaaa*. La version suivante fait le travail. Elle utilise la classe `std::ostringstream` qui permet de faire une écriture dans une `std::string`.

```
/*
 * Rôle : écrit l'entier n sur c caractères éventuellement
 *        précédé de 0
 */
static std::string intcChr(const int n, const int c) {
  std::ostringstream oss;
  oss << std::setw(c) << std::setfill('0') << n;
  return  oss.str();
}

std::string toString() const {
  return intcChr(this->jour,2) + "/" + intcChr(this->mois,2) +
                                 "/" + intcChr(this->année,4);
}
```

.....

▶ 4. Déclarez une variable `d` de type `date` initialisée à la date du jour (*i.e.* 26/10/2023).

```
date d(26, 10, 2022);
```

.....

▶ 5. Programmez la classe `personne` formée :
- de deux variables menbres privées : `nom`, de type `std::string`, et `naissance` de type `date` ;
- d'un constructeur public. Si la date de naissance n'est pas connue, on prendra *00/00/0000*.
- de la méthode `getNom` qui renvoie le nom de la personne courante ;
- de la méthode `setNaissance` qui associe une date de naissance à la personne courante ;
- de la méthode publique `toString` qui renvoie une chaîne de caractères au format :
  nom : jj/mm/aaaa
- de la surcharge de l'opérateur `<<` pour écrire une personne dans un `std::ostream`.

```
#pragma once

#include <string>
#include "date.hpp"

class personne {
private:
  std::string nom;
  date naissance;

public:
  personne(const std::string &p, const date &d=date(0,0,0)) :
    nom(p), naissance(d)
    {}

  std::string getNom() const {
    return this->nom;
  }

  void setNaissance(const date &d) {
```

```
    this->naissance = d;
  }

  std::string toString() const {
    return this->nom + " : " + this->naissance.toString();
  }

  friend std::ostream& operator<<(std::ostream &f, const personne  &p) {
    return f << p.toString();
  }
};
```

.....

▶ 6. Déclarez la variable `p` pour représenter la personne de nom *Danaé* née le *10 décembre 2003*.

```
personne p("Danaé", date(10, 12, 2003));
```

.....

▶ 7. Écrivez la déclaration de la variable `liste` de type `std::vector` de `personne` initialisée avec *Hadès* né le 30 mai 2000, *Héméra* née le 1er août 2002, et *Perséphone* née le 17 mars 2004.

```
std::vector<personne> liste = {
   personne("Hadès",  date(30, 05, 2000)),
   personne("Héméra", date(1,   8, 2002)),
   personne("Perséphone", date(17,  3, 2004))
};
```

.....

▶ 8. Écrivez la procédure `printListe` pour écrire sur la sortie standard les noms et les dates de naissances des personnes contenues dans un `std::vector` de `personne` passé en paramètre.

```
void printListe(const auto &l) {
  for (const auto &p : l)
    std::cout << p << std::endl;
}
```

.....

▶ 9. Écrivez la déclaration d'une variable `pp` qui sera initialisée à la valeur *toto, 26/10/2023* **allouée dynamiquement**. Affichez cette valeur sur la sortie standard, et enfin désallouez cette valeur.

```
personne *pp = new personne("toto", date(26, 10, 2023));
std::cout << *pp << std::endl;
delete pp;
```

.....