

Devoir écrit

Programmation en langage C++ L2/PL2 — TI130

Durée : 1h45

Sans document

Le barème est indicatif

Questions de cours (5 points)

Des réponses rédigées et argumentées sont attendues.

1. Qu'est-ce qu'une référence ? Donnez deux différences concrètes avec les pointeurs et deux cas d'utilisation.
2. Dans quel fichier se trouve traditionnellement la déclaration de la classe ? Les définitions de ses méthodes ?
3. Quels sont les rôles respectifs des constructeurs et destructeurs dans une classe ?
4. Que signifie « surcharger une fonction » en C++ ? Donnez un exemple.
5. Quel opérateur utilise-t-on pour créer un nouvel objet en mémoire de façon dynamique ? Quel opérateur pour le supprimer ?
6. Qu'est-ce qui différencie une relation d'agrégation d'une relation de composition entre deux objets ? Donnez un exemple.
7. Dans quelle(s) condition(s) la définition explicite d'un destructeur — différent du destructeur par défaut — est-elle nécessaire dans l'écriture d'une nouvelle classe ? Donnez un exemple.
8. Quelle(s) conséquence(s) y a-t-il à rendre un attribut d'instance « protected » au sein d'une classe ?
9. Pour quelle raison certaines méthodes sont définies comme « const » dans une classe ? Donnez un exemple.
10. Donner la signification d'une classe abstraite et pourquoi on ne peut pas l'instancier ?

Exercice (15 points)

I - Ecrire la classe abstraite nommée « *employe* » (5 pts)

- Tout employé a un nom et un prénom (par défaut, ça sera « DURANT Marie »)
- Tout employé a un numéro unique (identifiant de type entier, vous devrez utiliser un membre statique entier pour compter le nombre d'instances de cette classe)
- Tout employé a une qualité (ouvrier ou contremaître)
- Tout employé a des horaires de travail (créez une classe annexe *horaire* avec des champs heure/minute)
- Tout employé s'identifie par le biais d'une méthode *what()* dont voici le prototype et le code :

```
virtual string what() ; // prototype de la méthode what()
```

```

// definition de la méthode what()
string employe::what()
{
    string res = "nom_du_membre_stockant_le_nom" ;
    res += "nom_du_membre_stockant_le_prenom" ;
    res += "nom_du_membre_stockant_la_qualite" ;
    ostringstream _dum ;
    _dum << "nom_du_membre_stockant_l'identifiant" ;
    res += _dum.str() ;
    return res ;
}

```

- L'opérateur de sortie `operator<<` doit être surchargé.
- Tout employé peut (et doit) travailler en fonction de son poste, mais on ne sait pas comment : vous utiliserez une méthode virtuelle pure.
- N'oubliez pas d'implémenter les constructeurs / destructeurs

II - Ecrire une deuxième classe *salaire* (2 pts)

Elle représente le salaire d'un employé (salaire brut et taux de charges salariales). Vous pouvez utiliser des attributs « public » de manière à utiliser cette classe comme une structure.

III - Définir les classes *ouvrier* et *contremaître* (5 pts)

Il y a de manière claire une relation d'héritage entre la classe *employe* et ces classes. Les contraintes exigées pour ces classes sont les suivantes :

- Un ouvrier a un horaire d'embauche et quitte son travail 8h plus tard ;
- Un contremaître pilote plusieurs ouvriers : il faut créer une association, la classe *contremaître* doit stocker plusieurs *ouvrier* ou *ouvrier* * (à vous de choisir).
- Un contremaître a comme horaires de travail des horaires lui permettant de toujours être présent avec les ouvriers qu'il supervise.
- Pour travailler un ouvrier fait du « travail à la chaîne » (la méthode *travailler* retourne simplement la chaîne de caractères)
- Pour travailler un contremaître « supervise les ouvriers » (la méthode *travailler* retourne simplement la chaîne de caractères)
- Un contremaître reçoit une part de salaire variable en fonction du nombre d'ouvriers qu'il pilote (à vous de décider combien)
- Les ouvriers touchent un SMIC (Salaire Minimum pour Implémenter du C++) brut de 1800 euros, avec un taux de charges salariales de 15%
- Les contremaîtres touchent un fixe de 2500 euros avec un taux de charges de 20%.
- Vous écrirez les constructeurs / destructeur qui vous semblent nécessaires pour ces classes.

IV - Ecriture du main (3 pts):

Vous créerez une entreprise qui sera un conteneur d'employé : vous choisirez le type de conteneur que vous utiliserez (depuis le simple tableau statique jusqu'aux conteneurs de type `std::vector` ou `std::set`) Les fonctionnalités suivantes devront être implémentées :

- Ajouter un employé à l'entreprise
- Créer toutes les fiches de paie de manière simple : une fiche de paie comporte le nom, prénom et qualité de l'employé (ouvrier ou contremaître), son identifiant, et le montant de sa paie : on indique le salaire brut, les retenues (taux de charge * salaire brut) et le salaire net (salaire brut - retenues)