Devoir écrit
Programmation en C++ (I) — TI133P

Durée 1h45
Sans document
Le barême est indicatif

Questions de cours (5 points)

Des réponses rédigées et argumentées sont attendues.

1. Quelle est la fonction du compilateur ? De l'éditeur de liens ?
2. Qu'est-ce qu'une référence ? Donnez deux différences concrètes avec les pointeurs et deux cas d'utilisation.
3. Comment concrétise-t-on l'état et le comportement d'un objet dans la déclaration d'une classe en C++ ? Donnez un exemple.
4. Dans quel fichier se trouve traditionnellement la déclaration de la classe ? Les définitions de ses méthodes ?
5. Que signifie « surcharger une fonction » en C++ ? Donnez un exemple.
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. À quoi sert une surcharge d'opérateur en C++ ? Il existe deux manières de mettre en place une sucharge d'opérateur. Expliquez lesquelles et Donnez des exemples pour illustrer vos propos.

Exercice 1 (8 points)

Analyse et modification de classe (4 points)

On souhaite écrire une classe, nommée `DispositifEcriture`, permettant de mettre en œuvre un dispositif virtuel d'écriture.

```
#include <iostream>
using namespace std;

class DispositifEcriture {
    double  _prix;
    string  _marque;
public:
    string  getMarque() const;
    void    setPrix(double prix);

    void    ecrit(const string& message);
};
```

Sur votre copie, reprenez la déclaration de cette classe en y ajoutant :

- les accesseurs et mutateurs nécessaires pour tous les attributs existants ;
- un constructeur dont le paramètre permettant d'initialiser la marque du dispositif a une valeur par défaut égale à `"Generique"` et le paramètre permettant d'initialiser le prix une valeur par défaut égale à `0`.

Que peut-on dire de ce constructeur ?

La méthode `ecrit(...)` a pour rôle d'afficher un message à l'écran. Complétez la classe `DispositifEcriture` en écrivant les définitions de toutes ses méthodes dans un fichier source (ne pas oublier l'inclusion du ou des fichiers nécessaires).

Conception de classe (4 points)

On voudrait maintenant spécialiser la classe `DispositifEcriture` de façon à pouvoir représenter un dispositif nommé `DispositifEcritureMajuscule` qui a les mêmes caractéristiques que `DispositifEcriture` à ceci près que sa méthode `ecrit(...)` affiche les messages entièrement en majuscule.

Comment peut-on créer cette nouvelle classe en s'appuyant sur la classe `DispositifEcriture` qui existe déjà ?

On suppose qu'il existe une fonction interne (méthode) de la classe `string`, nommée `toupper()`, sans paramètre et qui retourne une chaîne de caractère de type `string` contenant une copie de la chaîne de caractère initiale en majuscule.

Écrivez la classe `DispositifEcriture` dans un fichier en-tête et un fichier source appropriés. N'oubliez pas de bien définir le constructeur et les redéfinitions de fonctions éventuelles.

Écrivez un programme principal qui met en œuvre les deux classes définies (attention aux inclusions de fichiers en-têtes) en illustrant bien leur différence. Précisez ce qui apparaît en sortie de l'exécution de votre programme.

Exercice 2 (7 points)

Le but est de créer une classe `GInt` de très grands nombres entiers naturels de taille illimitée (sauf par la mémoire disponible). Cette classe doit contenir un seul attribut privé uniquement : un tableau dynamique de caractères de type `string`.

Chaque caractère de la chaîne de caractères représentera un chiffre (entre `'0'` et `'9'`) du grand nombre entier correspondant.

On veut pouvoir :

- initialiser/affecter un grand entier à partir d'une chaîne de caractères, d'un nombre entier (de type int), ou d'un autre grand entier ;
- afficher un grand entier via `cout` en surchargeant l'opérateur de sortie << ;
- additionner et multiplier un grand entier à un autre via la surcharge de l'opérateur binaire habituel d'addition +.

Voici un exemple de programme principal (main.cpp) que l'on doit pouvoir écrire en utilisant la classe `GInt` :

```
#include "gint.h"
void main() {
    GInt a(576), b("76245"), c;

    c = a + b;
    cout << c << endl;
}
```

On rappelle qu'on peut utiliser l'opérateur + avec des objets de type `string` mais que cette opération réalise la concaténation des deux chaînes de caractères et non leur somme entière, ainsi `"576" + "76245"` donne `"57676245"` et non `"76821"`. On précise également qu'il est possible de concaténer un caractère de type char `c` à une chaîne `s` de type `string` via l'opération `s+=c`. Par ailleurs, la méthode `s.length()` de la classe `string` permet d'obtenir la longueur de la chaîne de caractères `s`.

Pour ajouter deux grands entiers représentés par des chaînes de caractères, il faut être capable de combiner les chaînes `"576"` et `"76245"` en appliquant les règles classiques de calcul d'une addition chiffre par chiffre (attention aux retenues et à l'ordre de traitement des caractères). Notez qu'on peut évaluer la valeur entière d'un caractère en lui retirant le caractère `'0'`. Ainsi, l'expression `(int)('6'-'0')` vaut 6.

Écrivez l'interface de la classe `GInt` dans un fichier en-tête nommé `gint.h` et la définition de la classe `GInt` dans un fichier source nommé `gint.cpp`. de façon à ce que le programme ci dessus affiche bien `76821`.

Exercice bonus

Adaptez la classe `GInt` pour manipuler les entiers relatifs et non seulement les entiers relatifs.